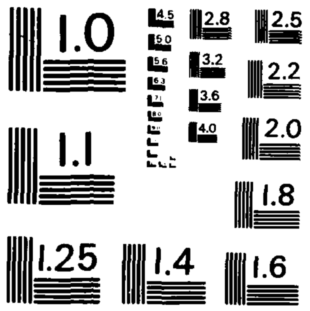
AD-A122 743 JOURNAL OF COMPUTERS (SELECTED ARTICLES)(U) FOREIGN TECHNOLOGY DIV WRIGHT-PATTERSON AFB OH G LIU ET AL. 14 OCT 82 FTD-ID(RS)T-0369-82 1/2

UNCLASSIFIED F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

②

AD A122743

FTD-ID(RS)T-0369-82

FOREIGN TECHNOLOGY DIVISION

AIR FORCE SYSTEMS COMMAND

JOURNAL OF COMPUTERS

(Selected Articles)

FOREIGN TECHNOLOGY DIVISION

82 12 28 131

FTD-ID(RS)T-0369-82

EDITED TRANSLATION

FTD-ID(RS)T-0369-82 14 October 1982

MICROFICHE NR: FTD-82-C-001317

JOURNAL OF COMPUTERS (Selected Articles)

English pages: 101

Source: Jisuanji Xuebao, Vol. 4, Nr. 5, 1981, pp. 321-371

Country of origin: China
Translated by: SCITRAN
F33657-81-D-0263
Requester: FTD/TQTA
Approved for public release; distribution unlimited.

PREPARED BY:

TRANSLATION DIVISION
FOREIGN TECHNOLOGY DIVISION
WP-AFB, OHIO.

TABLE OF CONTENTS

CZXT-013 AND SOME PROBLEMS IN ITS IMPLEMENTATION

Liu Guo-Heng

Institute of Computing Technology,
Academia Sinica

ABSTRACT

In this paper, the function and structure of CZXT-013, which is an operating system on the computer 013, are described. Some problems in CZXT-013 design and implementation, such as resource management, process communication and so on, are discussed. /321*

The resident mainframe 013 Computer is a medium-sized general purpose computer suitable for scientific computations. It can process approximately 2 million instructions per second and is equipped with a dynamic page address translation mechanism. The main memory consists of 128 K words with 48 bit word length. Its byte multiplexer channel and block multiplexer channel are connected to normal I/O devices and magnetic disk drives respectively. There are 4 disk drives with a capacity of 850 K words per drive, the average access time being 800 millisecond.

Taking into consideration our technical level and experience in operating systems research as well as the current status of our computer hardware, we have decided upon a design principle of being practical, plain, reliable, efficient and with room to spare. Under the premises of satisfying the general requirement of scientific computation, we adequately controlled the scale of the system and made it into a reliable and practical system.

*Numbers in margin indicate pagination of original foreign text.

Received 5 August 1980.

I. GENERAL FEATURES OF THE SYSTEM

The CZXT-013 is a multi-programming batch processing operating system oriented for scientific computations. It can support 5 detached and 2 online jobs simultaneously. In order to correlate the jobs so as to utilize the system resources in a balanced and highly efficient way, the system is scheduled on 3 levels. Job scheduling is high level scheduling which selects suitable jobs from the job queue for processing. On the medium level, there is main storage scheduling which determines dynamically for the various jobs the number of pages in the main memory. On the low level is process-scheduling, which distributes the CPU to the various processes. The system provides independent virtual storage page space to each job in the operating state. The maximum address space of each job is 1 million words. Each page contains 512 words and may be saved or deleted by the page from the main memory. While designing the system, we foresaw the problem of speed mis-match between the ability of the disk drive to provide page space and the demand on page space by the CPU during data processing, but we still decided to adopt the virtual storage method so as to learn how to manage a virtual storage system. However, we have incorporated certain measures to reduce the speed mis-match problem to our best ability. In order to shorten the job I/O waiting time, increase the job I/O flexibility and decrease the number of I/O devices, we used a pseudo-offline mode of I/O operation in the system. File processing is also provided to facilitate user data access. In view of the limitations of the external storage devices, only some basic functions are provided in the file system. 32 logic devices are provided by the system to the user so that the user may conveniently utilize the files and external devices and so that /322 user program will be sufficiently independent of the files and actual devices. In user programs, read and write to files and peripheral devices are treated as read/write to certain logic devices, and the correspondence between logic devices and the files or peripheral devices may be delayed until job execution time when it will be established through user instructions.

This system is one oriented toward scientific computations with primarily batch processing; hence the job control language provided by the system emphasizes the special characteristics of scientific computation and the need for organizing operational documentation. To lighten the load of online user operation (currently, our users are not yet familiar with keyboard operation), the system allows online jobs to have operation documentation. An online user may choose between manual or automatic operation for job processing as well as mixing the two modes. In order to provide relatively complete online service for the users while avoiding excessively lengthy programs explaining the commands, a service program library is established in the system, which also facilitates the augmentation of system functions.

Three concepts are employed, or rather 3 design tools are used, in the structural design of the system, i.e., hierarchy, module and process. The employment of these 3 concepts makes the system structure clear, reliable and orderly.

In order to realize the full function of the system, we augmented CZXT-013 altogether 8 times. The augmented programs on each occasion form one heirarchy, hence the structure of the system consists of 8 hierarchies. Some of the hierarchies are further divided into grades. There is a semi-ordering uni-directional dependence among the hierarchies. The system programs within each hierarchy are further divided into modules with unique and precise functions. A uniform format for calling and communicating among the program modules is also stipulated. The modules in the kernel hierarchy are realized in the primary language. Calling of the primary language is realized through the "transfer" instruction. The messages are transfered through the accumulator. The modules in the other hierarchies operate as processes. Processes interact with the exterior through calling the primary language provided by the kernel hierarchy. One important characteristic of an operating system is its concurrence. The concept of "process" is introduced to emphasize the concurrence of CZXT-013. The processes in the system are divided into one system family and several job families. Job family processes are a group of processes active around a job in operation. They appear and disappear with the creation and deletion of the job. The number of job families is the same as the number of jobs in operation status.

In order to facilitate system operation and to reduce the total number of processes in the system, a family of permanent system processes is established when the system is first turned on. This family of processes is oriented toward the whole system and does not serve just one job. The co-operative relationship between processes is realized through process communication. The communication format in our system consists of signal volume operation and message buffer, both realized through the related primary language provided by the kernel hierarchy. The unified control of communication makes it easier to manage and test the communication between processes, hence making it more reliable and secure.

II. A FEW PROBLEMS IN PRACTICE

Quite a few problems were encountered during the design of the CZXT-013. Some design loopholes were also discovered during the actualization process. Some of these problems have been successfully solved by us, but some of the solutions have not been satisfactory. In the following, we shall discuss some of the problems in resource management and process communication and introduce some of our solutions.

1. Centralized management of shared resources

We know that the main purpose of designing an operating system from the viewpoint of resource management is to share the computer among many users so that the resources of the computer system may be utilized very efficiently. As pointed out by P. Brinch Hansen: "There is an economical necessity in the sharing of computer equipment and the purpose of the operating system is to make this sharing a practical reality." The first thing encountered in the realization of resource sharing is how to manage the shared resources. The most natural and effective method for the computer as a data processing tool to manage shared resources is to treat the shared resources abstractly as a set of data. That is, to represent the resources as data and to reduce the management of shared resources such as distribution, retrieval, etc., to a set of operations on these resource /323 data. The processes of shared resources must operate separately on this set of data.

When a distributed method of management is used, in order to guarantee the uniformity and integrity of resource data, each process is required to operate on the resource data during the corresponding exclusive steps. This not only makes it necessary for the resource sharer to go through elaborate operating procedures, but also introduces many undesirable factors into the system: First of all, it is difficult to guarantee the reliability of the system. If only one sharer operates unsuitably on the resource data, the normal operation of the whole system will be affected. Secondly, it is inconvenient to scrutinize the "privilege" of the sharer, thus affecting again the security of the system. Furthermore, by exposing the resource data in front of each sharer and letting them operate on the resource data individually, it becomes very difficult to modify the resource data, thus damaging the maintainability of the system. Therefore, a centralized management method is used in CZXT-013 for managing shared resources: only a special manager is responsible for maintaining the resource data of sharable resources. It provides the sharer the necessary service operations. The operations of the sharer can only be carried out by the manager. As an example, this centralized management method is used in such sharable resources as the main memory storage, the magnetic disk drive and I/O ports. This has not only improved the reliability, security and maintainability of the system, but also simplified the system programs.

Although the structural concept of "management program" is not explicitly used in the management of sharable resources, the thinking behind this type of centralized management of sharable resources is basically the same as that behind management programs. It helps to improve the function of the system.

2. Prevention of deadlock in resource management

In order to improve the rate of resource utilization, the dynamic allocation method is often used in resource management. A deadlock may occur when dynamic allocation is applied to resources which cannot serve several processes simultaneously and which cannot be retrieved by force by the system. Only when the amount of resources is sufficiently large or when the number of requests by jobs or processes is limited can we avoid the deadlocks produced by severe competition. The number of system resources (such as hardware) is limited, and can never be sufficiently large for a multi-programming system. It is often un-economical and impractical to increase the amount of resources to the level of being sufficiently large. Thus,prevention of deadlock should not be neglected when resource allocation strategy is being considered. There may be many ways to define deadlock, but we consider the following to be adequate: "Deadlock is a state in which two or more processes wait indefinitely for a condition that can never be realized." The necessary conditions to produce deadlock in resource management may be summarized as:

(1) A certain part of the resource can only be used by one process, i.e., its utilization is exclusive.

(2) The resource cannot be scheduled in a non-deprivable way, i.e., only the user can release the resource which cannot be retrieved forcibly by t e system

(3) A group of processes wait for the resource cyclicly, i.e., some processes in this group have already occupied part of the resource and they are waiting for the release of the resource by other processes so that they may obtain the remaining resource.

The resource allocation strategy of CZXT-013 is determined according to this kind of analysis. Management of the main memory is divided into the page allocation method and the non-page allocation method. Their objects of management do not overlap. Among these, the main storage pages (at least most of them) may be retrieved by force and therefore deadlock will not be produced because of competition for them. The non-page allocated main storage is non-deprivable, and hence the first 2 necessary conditions both hold. To prevent deadlock, we must make sure that condition (3) must fail to hold. The most convenient method to destroy condition (3) is to avoid having the requesting process wait for the resource. However, this must result in increasing the amount of non-page allocated main storage, so that the need for procedure control blocks, procedure work space and message buffer may be amply satisfied.

When there are few jobs being executed and only a small number of processes, the main storage will be wasted. We have adopted a method of modifying the character of resource allocation to avoid both deadlock and waste of main storage. When a job is being established, a part of the requested main storage pages is used as the control block and work space for this family of processes, thus decreasing the amount of non-page storage requested. Although this part of the main storage pages cannot be forcibly retrieved by the system, it is small compared to the total number of pages that can be forcibly retrieved (e.g. main storage pages supporting vitual storage). Also,there are special programs in /324 the system to control the number of pages in main storage used for virtual storage; hence,deadlock will not develop due to page allocation. Thus,other than the fixed number of storage units used to establish the procedure control blocks and work area of system procedure family as long as the buffer units needed by the message buffer can be guaranteed. We may consider the storage to be sufficiently plentiful and no deadlock will occur. Very little main storage is used by the message buffer; only 4 units at a time. In this way, only a small price needs to be paid to avoid the deadlock arising from competition for main storage.

The allocation characteristic of the disk sector area used for I/O port also satisfies the first 2 necessary conditions for producing deadlock, but the number of port areas is sufficiently large to satisfy the need for normal operation of the jobs so no deadlock will occur. However, system deadlock may still be caused if a detached job outputs and requests for disk sector areas indefinitely. For this, we adopt the method of limiting the actual length of port file. When a detached file exceeds the allowed length (256 sectors), the system will automatically output it in batches while retrieving the port areas for other processes to use.

3. Prevention of loss of resources

We discover that a problem worthy of attention during the system realization process is the prevention of loss of resources. Commonly in resource allocation, the system usually only records unallocated resources while users of resources already allocated are no longer recorded. This is especially true when the number of resources is large and users are many. A loophole may thus be created when a process fails to release voluntarily its resources during an abnormal termination. A more serious situation arises when a procedure is forced to terminate after its requested resources are allocated, but before they can be recorded in the appropriate data base. A loss of resource will result in both cases. This problem must be properly treated in order to make the system less vulnerable and more error-tolerant when the mainframe's reliability is low.

To prevent resource loss by recording the requesting processes during allocation will result in wasting large amounts of space and increasing the overhead of the system to an intolerable degree. An economical and simple method must be formed to prevent the loss of resources. We have not yet found a unified and general solution during the process of treating this problem. Only some specific measures for solving practical problems have been adopted. Registration of allocated resources into tables of the same form will be done by the allocator; temporary registration is established by the system in the PCB of the process when tables of different forms may be involved. Because of series of measures taken by us, we have basically insured the complete retrieval of system resources whether the job ends normally or abnormally.

4. Problems of process communication

Another problem investigated by us while programming the CZXT-013 is the effect of process communication on system security. As mentioned above, we introduced processes in the structural design of the system, and classified processes into one system family and several job families. The control and entrusting relationship between processes is realized through process communication which takes place not only among processes within the family, but also between the system family and a job family.

Two means of communication are provided by the system. One is a low level primary language communication for signal volume operation. This is an implicit communication among processes,as the operator does not know clearly the dialogist; the other is a high level primary language for communication message buffer which is explicit communication with the operator knowing clearly the dialogist. The former is simple to realize, has high efficiency, low message volume and poorer security, while the latter has high message volume, more security, but higher overhead. No problem will arise during normal system operation as long as the logical relationship in the design is correct. However, once an accidental event occurs in a job, causing the abnormal cancellation of this job family, the system may be paralyzed due to the destruction of a normal communication relationship. For example, if a procedure is cancelled after a P operation on a public, exclusive signal, but before a V operation, then other operators may be indefinitely blocked. Similarly, in high level communication, there may also be phenomena of one or several processes waiting indefinitely for messages from a cancelled process. To overcome /325 this deficiency, we imposed some limitations on process communication and also made some extensions to the communication.

Firstly, we limited the use of signals. Signals can only be used within the family. No public signal between different families is allowed. This not only capitalizes the advantage of signal communication within limits, but also avoids its possible problems. One point needs to be clarified. A public signal may be used by the kernel hierachical program, because this problem has been considered in system design. The asymmetry thus produced may be avoided because (SMC segment) "System must complete" segment is installed. Furthermore, when an error occurs in the kernel hierarchy, the error level is higher than those which occur in non-kernel hierarchy programs. The method of treatment will also be different.

Secondly, we imposed some restrictions on the use of message buffer communication and also added some extensions to the means of communication. When communications which require replies occur between processes of different families, the system will make a record. When the process for the job family is cancelled, if the process of the system family has not yet issued a reply, then the cancellation will be delayed until such time as a reply is received. If the job family process has not issued any reply to the system family process, then a "remedial" message will be issued by its parent process. This method is used to overcome the asymmetry of communication so as to avoid the damage that an abnormally-terminated job may bring to the system.

III. CONCLUSIONS

After adjustment and testing, CZXT-013 has now entered the usage stage. Test results indicate that the original design goals have basically been achieved and that the system functions can satisfy the general requirement for scientific computations and operate stably. The system should require further improvement. Currently, an important factor affecting the operation of the system with high efficiency is hardware stability, disk volume and disk speed. This problem is being solved.

CZXT-013 is researched and developed by the Computing Technology Institute of the Academia Sinica. Several comrades from the 9th Institute of Beijing also participated in this work, which has also been supported by some comrades from the Computing Center of the Academia Sinica.

CZXT-013: A HIERARCHICAL OPERATING SYSTEM

Zhang You-La   Jia Yao-Mei

(Institute of Computing Technology, Academia Sinica)

Abstract

CZXT-013 is a batch-processing operating system, capable of handling eight jobs simultaneously. This paper describes the architecture design for the system, i.e., the design objectives and the tools (hierarchy, process and modularity) which are needed to realize the objectives. The kernel primitives and their design considerations are illustrated in detail, the function of each process, process control and communication are also described. The experiences gained in the course of design are presented at the end.

I. Introduction

CZXT-013 is a batch processing operating system. It allows at most 8 jobs to run simultaneously in the computer. The 8 jobs are: job 0 is a system job. It is responsible for managing the system's total input/output (I/O) devices, the internal storage scheduling and entering of jobs; job 1 is online service job, which enables the user on the monitoring console to request for system service through "service" commands, such as writing file marks on tape sections, writing file headers on tape, compiling source programs, etc.; job 2 is an online job which enables the user on the console to control the execution of his job through a series of online commands; jobs 3 through 7 are batch processing jobs which run jobs through the operation document.

In order to improve reliability and flexibility, 3 structural design tools have been used in CZXT-013. The first is process; the

Received August 5, 1980

second is modularization which means that the system program is divided into many modules, each of which serves a particular function. The modules can be individually programmed and tested. Communications among the modules follow a uniform interface. The third is hierarchy. The modules and processes are divided into several unidirectionally dependent hierarchies according to their functions.

The common feature of the three tools mentioned above is the localization of global problems. However, each tool achieves localization from a different angle. The process divides the complete process of CPU activities into many small independent units; modularization divides a large, complicated program into many independent, small modules; hierarchy divides the whole system into several unidirectionally dependent hierarchies. In this way, to understand the whole system is to understand each process, each module and each hierarchy and to program and test the whole system is to program and test each process, each module and each hierarchy. This makes it easier to understand the whole system, more convenient to test it, and also easier to make local modifications. Besides, loops are reduced by the hierarchical structure and so is the number of errors, hence the improvement in reliability.

We divide CZXT-013 into 8 layers of hierarchy. The 0 layer is the central layer, called the kernel. The other 7 layers are processes. The kernel and each process are sub-divided into modules. We shall describe their individual functions and installation below.

II. The Kernel /327

The kernel is the bottom-most heirarchy of the system. It is the direct extension of the "base machine". Its function is to provide a good environment and create the necessary conditions for the normal operation of the processes, which means that the process runs on the virtual machine extended by the kernel. The kernel consists of some primitives and interrupt processing programs.

1. <u>The sub-divisions of the kernel and their functions</u>.

The O/H hierarchy formed by the kernel is sub-divided into 9 sub-divisions, as shown in Figure 1.

Level 0		XTIMHG (high frequency clock)		
Level 1		XCPUSI (CPU scheduling)		
Level 2	XP (P operation)	XV (V operation)	XVV (broadcast V operation)	
	XENTRY (SMC entry)	XEXIT (SMC exit)	XW (W operation)	XN (N operation)
Level 3		XB (FSB chain)		
	XALOC (non-page allocation)	XFREE (non-page region release)	XG (page allocation)	XJ (page release)
	XGP (page table request)		XJP (page table release)	
Level 4	XCREAT (create)	XSTART (start)	XSTOP (stop)	XDESTR (destroy)
	XSEND (send message)	XSENK (send kernel)	XSENDF (send family)	XSENWA (send and wait)
	XANSR (send answer)	XANSRF (answer family)	XRECEI (receive message)	XWAIT (wait)
		XQUE (error report)		
Level 5		XTIMLW (low frequency clock)		

```
Level 6                         XIOCS
                         (I/O start control)

Level 7    XL          XM         XK        XH        XU       XSAVE
         (address   (monitor)  (trace)   (hang)   (unlock   (save
          check)                                    process) status)

Level 8            INTERO                    INTER1
             (0 level interrupt         (1 level interrupt
              processing)                processing)
```

Figure 1. Kernel subdivision.

Level 0 is the high frequency clock primitive XTIMHG. It is responsible for managing the hardware fast clock. The fast clock after XTIMHG extension is capable of implementing the process time slice.

Level 1 is the low level scheduling primitive XCPUSC. It is responsible for allocating the CPU to each process. Through XCPUSC, CPU becomes many virtual CPU's so that each process will regard itself as processing a CPU.

Level 2 is the low level communication primitive and the processing of the "System Must Complete segment" (SMC segment). The treatment of SMC segment will be introduced at the end of this /328
section, and the design background and usage of the low level communications primitive will be introduced in the next section.

Level 3 is the allocation of real storage other than the fixed storage, the rest of the 130K main storage of Model 013 computer is the dynamic user area. This is divided into paged region and non-paged region according to usage. The paged region is allocated by page (each page 512 words). It is used in user program data area, system I/O buffer area, etc. The non-paged region is allocated according to number of units. It is used for process control block

(PCB), message buffer, process work space, etc. All free non-paged storage blocks (FSB) are linked together as an FSB chain. Process requests for a non-paged area pass through the XALOC primitive. The primitive finds a large enough FSB from the FSB chain and allocates it to the requester. The release of the non-paged area is done by the XFREE primitive. The request and release of the paged area are done by XG and XJ primitives. We have also set up the XGP and XJP primitives to provide the reservation and release of designated pages. When the high level process has chosen a job for execution, and needed page table area to allocate virtual space for the job (page table area position is fixed), if the page in the page table area is already occupied, then the high level process must reserve the needed pages, and release them through XJP. After the extension of this level, the internal storage becomes sufficiently numerous to be used by the process.

Level 4 are the process control and high level communication primitives. The design background of these primitives will be introduced in the next section.

Level 5 is the low frequency clock primitive XTIMLW. This primitive extends the hardware slow clock to a job alarm clock (to record job run time), a process alarm clock (to record user process run time), a delay alarm clock (to resume operation after sleeping for so many seconds), etc. to satisfy the timing needs.

Level 6 is the I/O start-up control primitive XIOCS for starting, stopping and checking the exchange channels and disk channels. External device channels then become logic channels after this extension.

Level 7 is the scheduling primitive. It facilitates user scheduling of processes, such as the XL primitive for processing address check, the XM primitive for monitoring, the XK primitive for tracing, etc.

Level 8 is interrupt management, namely the level 0 interrupt management INTER0 and Level 1 interrupt management INTER1. Its job is to manage interrupt or to convert the interrupt signals into messages and send them to the appropriate processes. After this level of extension, the concept of interrupt disappears from the hardware machines. Presented to the processes are then messages. The process understands the working conditions of the channels and other hardwares through the messages.

2. <u>The main consideration in kernel design</u>.

The kernel is the center of the system. Its efficiency has a deciding effect on the efficiency of the whole system. Its correctness is the key to the correctness of the whole system. Hence the structure of the kernel must be considered very carefully. Two important measures are adopted in the design of the kernel, namely modularization and the concurrence of the primitives. Three meanings are included in the term "modularization." First is the unity of function. The whole kernel is composed of many modules, each of which consists of only one function. Each module is either a single primitive or a interrupt processing program. Second is modular independence, i.e., each module may be independently programmed and tested. Each module has a set of general registers (accumulator, address modification register, PSW register, etc.) and no module of interface. The kernel primitives may be called not only by processes but high level primitives may also call any low level primitive, i.e., nested call is permitted. Calling of kernel primitives is made through the "advance manage" (AM) and "retreat manage" (RM) modules. These two modules are not listed in the hierarchy structure in Figure 1. Nested calling messages are all transmitted through 2 accumulators.

Apparently, modularization clarifies the structure, making it easy to understand, modify and test, but the overhead of the system, both in time and in space, is increased. More than a hundred instructions are dynamically executed each time the kernel calls the AM and RM modules. More importantly, in order to save the contents of the general registers during nested call, 4 status retention areas and a separate system status retention area must be opened in the PCB table of each process. This will extend the length of each PCB to 40 units.

Another important measure in kernel design is the introduction /329 of concurrence in the primitives. The actions of the kernel primitive may be seen as the extension of the actions of processes. Therefore the concurrence of the process is also extended into the kernel. For example, 2 processes may enter into the XALOC primitive simultaneously and request for internal storage simultaneously. Of course, not all primitives permit concurrent execution. Some primitives remain in serial operational, e.g., all the level 2 and level 6 primitives. The primitives are allowed to work under "unlocked" or "locked" condition to realize the concurrence or serial-ness. Introduction of concurrence into the kernel is advantageous in improving the concurrence of the whole system, especially in shortening the real time response time and in speedy response to interrupt under realtime conditions.

However, the problems that are produced due to concurrent operation of processed will also appear in the kernel. First of all, conflict may occur in the primitives when 2 processes enter the kernel primitive simultaneously. The method we use to solve this communication problem is to adopt P, V operations, in the same way that the direct communication between processes is solved. Another problem is that it is easy for the system to become insecure. For example, when a process requests for internal storage area through the XALOC primitive, as this primitive is in the process of modifying the free storage chain (access to the chain is guaranteed to be exclusive with P, V operations), the free storage chain will be locked forever if the process should be interrupted and then stopped or cancelled for some reason. The method used by this system to solve this problem is to call such segment an SMC segment (System Must Complete Segment), namely those program segments which will cause insecurity when the process is stopped or cancelled during execution. We set up an SMC flag in the PCB of each process. The flag is non-zero when SMC is entered and zero when completely exited. The process can only be cancelled when the flag is zero. XENTRY and XEXIT are the primitives to set and clear the SMC flag.

III. Processes

1. The hierarchical structure of processes

Hierarchies 1 through 7 of our system are all system-defined processes. These processes are arranged according to the principle of unidirectional dependence. The hierarchical structure of the process is shown in Figure 2.

Hierarchy layer 1 are device management processes. They manage respectively the various external devices provided by the 013 computer. "Console CRT" and "Error Print" processes report to the operator, system operating status from the console CRT and system printer respectively. "User CRT" process reports job execution status to online user on the user console CRT. "Disk manager i" process is responsible for managing the ith disk drive ($i = 0,\cdots,3$). "Disk controler" process optimizes the disks. "Port allocator" and "port retrieval" allocate and release the I/O parts. "Tape i" ($i = 0,1$) manages the ith magnetic tape drive. "Online Print i" and "Offline Print i" manage respectively the ith channel online and offline printer. "Online input" and "offline input" processes manage online and offline paper tape input. "Light pen display", "Electrostatic" and "Punch" processes manage respectively the corresponding device.

Hierarchy layer 2 is storage management. "Page manager" process is responsible for internal and external page exchange. "Central Scheduler" process decides on the allocation strategy of internal real pages.

Hierarchy layer 3 is file management. The task of the "Basic files" process is to transmit file messages between the buffer area and external media while "Read," "Write" and "File Supervisor" processes transmit file messages between the internal storage and the buffer area. The "file supervisor" process is provided to create file supervisor for jcbs. The file supervisor includes job name, job type, operational documentation, start time, status, etc. The "Catalog" process manages file catalog and operations on the catalog, such as creation, open, close, delete files, etc. The "header" process reads and writes file headers on magnetic tape. The "offline output controler" process controls the offline output of files.

Hierarchy layer 4 is user process (or subsystems such as compilers) and job control. The "user command interpreter" process interprets and executes user commands. The "job controler" process receives commands and processes various events of a job.

Heirarchy layer 5 is job creation and scheduling. "Paper tape header" and "job creation" processes input all the key-punched messages of an offline job (documentation, program and data) to the I/O part and place the job in a ready state. The "High level scheduler" determines whether to permit a user console inline job to enter the system as well as selecting a job from the offline job queue on disk for execution in the internal storage. Resources will be allocated to all jobs and retrieved when the job ends. /330

Hierarchy layer 6 is system job control. The "Console keyboard monitor" and "user keyboard monitor" send commands from the console keyboard and user keyboard to the relevant process. The "system control" and "Console command interpreter" processes are similar to

Hierarchy Layer 1	1	Console CRT Error Print User CRT	
	2	Disk Manager 0··· DM3 Port Allocator Port Retrieval Tape 0 Tape 1 Light Pen Display Disk Controller	
	3	Electrostatic Online Print 0 OP1 Offline Print 0 OFP1 Punch	
	4	Online Input Offline Input	
Hierarchy Layer 2	1		Page manager
	2	Central Scheduler	
Hierarchy Layer 3	1	Offline Output Controller Header	
	2	Basic Files Read Write File Supervisor Catalog	Basic Files Read Write File Supervisor Catalog
Hierarchy Layer 4			User command interpreter User and subsystems Job Controller
Hierarchy Layer 5		High Level Scheduler Job Creation Paper Tape Header	
Hierarchy Layer 6		Console Command Interpreter System Control Console Keyboard Monitor User Keyboard Monitor	
Hierarchy Layer 7		Start-up Process	

Figure 2. Hierarchical Structure of Processes

job control and user command interpreter in function. The only difference is that the latter are used for user jobs which the former are used for system jobs.

Hierarchy layer 7 is the "start-up process." It is the first process of the whole system. It enters an infinite loop after completing system start-up. When the CPU is idle, the system will enter into the "start-up process".

2. Process family and process control

CZXT-013 divides the processes into 8 families according to the maximum number of jobs allowed. All processes in the same family cooperate to complete the job. Processes in family 0 correspond to system jobs while the other families correspond to user jobs. The first family corresponds to service jobs, the second family to online jobs and the 3rd through 7th families to offline jobs. The system family consists of the processes in the left column of Figure 2. Each family of the user families consists of processes on the right.

"Start-up process" is the first process in the whole system. When the system begins operation, it enter this process first. This process completes the system start-up. In particular, it creates and starts all the processes in the 0th family. After that the system awaits for the entry of jobs. When the input port has a job queue or when an online job or service job requests for entry into the system and if the "high level scheduler" process examines the job and permits its operation, it will create and start-up the first process of the user family, i.e., the "job control" process. The "job control" process will then create and start-up the rest of /331 the processes of the user family. Thus,the processes of CZXT-013 contain 4 generations as shown in Figure 3. All the processes of the same user family (including the user process) work together to

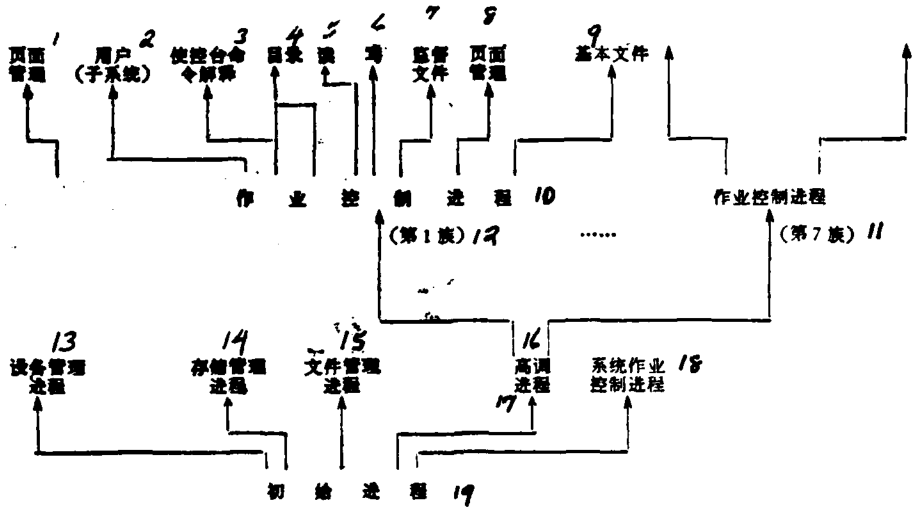


Figure 3. Process Families
Key: (1) Page Manager; (2) User (subsystem); (3) User console command interpreter; (4) Catalog; (5) Read; (6) Write; (7) File Supervisor; (8) Page Manager; (9) Basic Files; (10) Job Control Process; (11) Job Control Process (Family 7); (12) (The 1st Family); (13) Device Manager Process; (14) Storage Manager Process; (15) File Manager Process; (16) High Level Scheduler; (17) Process; (18) System Job Control Process; (19) Start-up Process.

complete the computational task of the job. When input and output are needed, messages will be sent to the device processes of the 0th family, requesting them to complete the I/O tasks. When a job ends, the "job control" process of its user family stops and cancels the other processes of its family and then the "job control" process is stopped or cancelled by the "high level scheduler" process. All the processes in the 0th family, including the "high level scheduler" are never cancelled. The 4 primitives XCREAT, XSTART, XSTOP and XDESTR in the kernel are used by the main process to create, start, stop and destroy sub-processes.

Device management processes are provided only in the 0th family so as to reduce the total number of processes in the system. If device processes are provided for each user family, then the total number of actual processes will greatly increase with the number of jobs in simultaneous operation, and the system overhead will be excessive.

3. Process states and their transition

The complete period of a process from start-up to deletion may be viewed as the life cycle of the process. We describe the stages of a process during its life cycle in four states. They are: stop, ready, execute and lock. After creation, the process is in the stop state. It is in the ready state after start-up. When a process in ready state is selected by the low level scheduler (XCPUSC), it then enters the execute state. When a process in execution cannot be executed temporarily due to some cause such as waiting for I/O to complete, it changes into the lock state. When the time slice of process execution is up, it also changes into the ready state. After the cause for locking is relieved, the process changes from the lock state to the ready state. No matter what state the process is in, it always turns into the stop state when it is stopped. The transition of status of a process is accomplished through the corresponding primitive. Figure 4 gives the 4 states, their transformations and the corresponding causes.

4. Process communication

In their direct, mutually governing relationship, the processes in a system also relate to one another parallelly in addition to the control relationship mentioned above (create and delete, etc.). Our system uses 2 kinds of tools to solve the parallel problem.

The first tool consists of the P, V operations. The XP, XV primitives in the kernel are used to implement respectively P and V operations. In addition, 3 signal-related operations are defined in CZXT-013 in accordance with practical necessity. The first is "broadcast" style V operation, namely V'(S) operation. The kernel /332 primitive XVV realizes this operation. Its difference from the regular V operation is that it awakens all the processes waiting at the corresponding signal. Sometimes several processes become locked because their requests for non-page storage blocks can not be satisfied. When the size of storage blocks which are released by some process, is larger than the total demand of all the locked processes, then the release action of this process should awaken all the locked processes. XVV is designed to achieve this goal. The other 2 signal-related operations are $P(S_1,S_2)$ and $V(S_1,S_2)$. The XW and XV in the kernel implement these 2 operations. $P(S_1,S_2)$ is equivalent to $V(S_2)$ and $P(S_1)$ operations with no interrupt allowed between them; $V(S_1,S_2)$ is equivalent to $V(S_2)$, and $V'(S_1)$ also with no interrupt allowed between them. $P(S_1, S_2)$ and $V(S_1,S_2)$ are mainly used in output process in critical regions. For example, the process segment in the kernel primitive XALOC, which examines whether the free storage chain (FSB chain) contains large enough FSB block is a critical region. P operation must be performed before the critical region is entered so that the other process can be locked out. When storage block of large enough size cannot be found after FSB chain has been searched, V operation must be performed before leaving the critical region to release the lock-out on other processes. P operation must also be performed on the waiting signal for internal storage to lock-out requestors. These 2 P, V operations before exiting from the critical region must not be interrupted, otherwise it will not be safe. Hence $P(S_1,S_2)$ is used to implement this demand. XFREE primitive uses the $V(S_1,S_2)$ operation after releasing the storage blocks prior to exiting from condition critical region.

The second communication tool is the message buffer. In the kernal, XSEND and XRECEI are the message sending and receiving primitives. Their meanings are the same as those of corresponding primitives in an ordinary system. To improve communication between processes, we have added to our system a few primitives. XWAIT waits for the message from some specific process and rejects

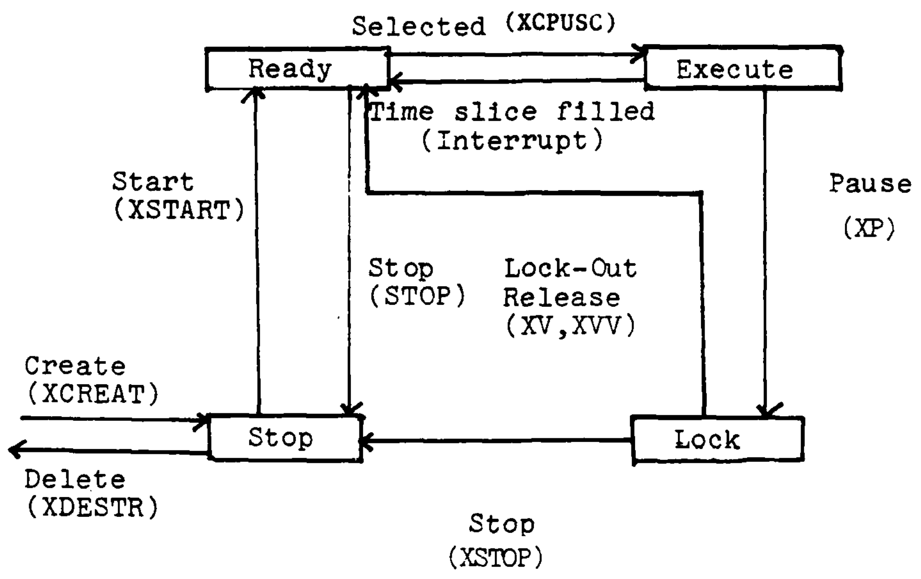


Figure 4. Transition of Process Status

temporarily the messages from other processes. The function of the XSENWA primitive is the sum of those of XSEND and XWAIT. It is easier to use XSENWA when the process waits for an answer immediately after sending a message. XANSR is answer dispatcher. A process sends message requesting service with XSEND. The server sends the answer message with XANSR after service.

As mentioned before, in order to reduce the total number of processes, device management processes are concentrated in the 0th family and not installed in each user family. However, the result is that communication relations must be established between each user family and the 0th family. Communication between different families involves more complications than that within the same family. For example, as the "basic file" process of an online user family sends a message to the "online print" process of the 0th family, asking for printer output but before the "online print" process replies, some accidental event causes the online user to end abnormally and the "basic file" process is deleted. The "online print" process has no way to know this. After it has finished printing and proceeds to send a reply, there is no one to receive the message. Because of this, 2 primitives XSENDF and XANSRF are established in the kernel to implement interfamily communication. Beside the functions of XSEND and XANSR, these 2 primitives are also implemented with inter-family communication management. An inter-family communication counter is installed in the PCB of each process. When the user family sends a message through XSENDF to the 0th family, this primitive increments the sender's counter by 1; when the 0 family replies through XANSRF, this primitive decrements the receiver's counter by 1. Only when the inter-family communication counter of a process is zero can it be deleted.

So far in our discussion of process communication, explicit communication operations, namely the communication primitives are involved, on both sides. There are also 2 rather special communication situations. One is the request for page allocation by a user process due to lack of page. Here the "page management" process has an explicit message reception operation by the user process and does not have any explicit message sending operation. Another is the communication between the external environment, such as the operator, and processes, e.g., the manual mounting of magnetic tape or paper tape by the operator. Here the external environment may be considered as an external process, and the situation then becomes that of the communication between an external process and an internal process. In CZXT-013, the 2 special communications are processed through interrupt. The lack of page space for a user process or the manual operation will both cause an interrupt. The interrupt manager transforms the interrupt signals into messages and sends them to the appropriate process. The XSENK primitive is provided for this purpose. It is called by the interrupt process to send messages for the user process or the external process. Both XSENK and XSEND are used to send messages. The difference is that the message buffer of the former is fixed, requiring no requesting since the interrupt process does not permit waiting for unavailable buffer for any length of time, while the message area of the latter is requested temporarily by XALOC.

IV. System Protection and Security

By system protection we mean than the hardware resources and software resources (programs, data bank) can only be visited by certain permitted modules or processes through clearly defined operations ,so that the resources will not be damaged or lost. In CZXT-013, aside from using such machine states as the virtual/ real states and computer/manage states provided by the hardware to localize errors, certain measures are also taken in the software.

To avoid damaging the system by its own errors, and to prevent the spreading of the effects of system software errors, we designed a supervisor system in the system. It is composed of several supervisor programs. These programs are inserted in the system programs at appropriate locations to supervise the correctness of system operation. For example, the supervisor programs inserted in the XFREE and XJ primitives primarily supervise the paged and non-paged release of the storage.

To improve resource utilization rate, dynamic resource allocation and usage strategies are used in CZXT-013. However, this caused problems in resource retrieval. In particular, when a job ends abnormally due to some accident, special care must be exercised in retrieving the resources allocated to the job, otherwise resource loss might ensue, causing the system to be unsafe. In the design of CZXT-013, each possible case for loss has been carefully analyzed and suitable measures have been taken. The treatment of SMC segment is a way to prevent the loss of the soft resource of shared kernel data bank. To prevent the non-page storage blocks requested by the process to be lost during abnormal deletion of the process, we link together the various storage blocks of a process. There is no pointer in the PCB process pointing to this claim. When the process is deleted, the XDESTR primitive inspects the chain and returns those storage blocks not released by the process itself.

V. Experience and Lessons

After more than 40 man-years of research and development, debugging, testing, and trial run over some period of time, the CZXT-013 hierarchical structured operating system basically achieved the expected result.

The system structure is successful. Due to the adoption of the design tools of hierarchy, process and modularization, the system structure is clear, which doubtlessly contributes toward the reliability of the system. Preliminary statistics indicate that the system overhead of CPU time (system program execution time and idling time) is generally 12-24% with an average of 18%. For jobs not as well fitted, the overhead may be somewhat higher. If jobs are purposely fitted, i.e., according to their volumes of computation, input/output, required virtual storage, etc., better efficiency may be achieved. Overhead for internal storage does not exceed 20%. Further analysis of the effect of system overall structure should only be done after the system has actually operated over a much longer period of time. At present we shall only discuss the following aspects:

1. The modular design of the kernel makes it highly reliable, easy to program, debug and test. Adding and deleting the primitives becomes very easy. But such a kernel design method with unitized function, modular independence and unified interface leads to the installation of 4 status reserve areas in the PCB of each process, thus increasing the overhead. On balance, we think this design is still acceptable.

2. The hierarchical structure of the processes clarifies the system structure and function, making it easier to understand and modify. But in the implementation at later stages, we did not strictly adhere to the principle of uni-directional dependence between the layers. Most of the processes are uni-directionally dependent but individual loops still exist, causing some software errors. For example, in the communication process, the phenomena of partly switching had occurred due to message loops. When Process A is waiting for a reply after sending a message to process B through the XSENWA primitive, it happens that process B also sends a message (not a reply) to process A. Process A accepts the service request message from process A as the reply to its own message and results in an error.

3. Concentrating the device management in the 0th layer and not spreading it in user families has no doubt reduced the number of system processes effectively. Optimization of the magnetic disk access can also be more easily implemented. However, some problems have also been introduced; e.g., the problem of inter-family communication. 2 primitives are added in the kernel to handle inter-family communication, making the system more complicated. Furthermore, this concentration is disadvantageous to the protection of the system. The jam phenomena occurred in the operation of the device management processes of the 0 layer will make the input/output operations of the user jobs in the other 7 families unable to proceed normally.

4. Although introduction of concurrence in the primitives has definite advantage in improving system efficiency, yet it has also greatly increased the complexity of the kernel. The communication problems and the treatment of SMC segment in the kernel are all caused by this. If the primitives are restricted to serial calling only, then both of these treatments may be eliminated. This not only will greatly simplify the kernel, but also will save all the CPU time spent on these operations. In a batch processing system

with no severe demand on response speed such as CZXT-013, introduction of concurrence loses more than it gains.

Comrades Yang Li-zhong, Chan Yu-yan, He Wan-hua, Liu Ming-di, Huang Wan-yong of the Computing Center, Academia Sinica, participated in the implementation of the kernel. We would like to extend to them our gratitude.

THE DESIGN OF THE CZXT-013 JCL (JOB CONTROL LANGUAGE) AND THE IMPLEMENTATION OF JOB CONTROL

Chan Hua-Ying   Lu Yu-zhen

Computing Technology Research Institute, Academia Sinica

Abstract

This paper describes a user-oriented job control language CZXT-013 JCL. The design principle and some specific commands are presented. Some key problems in realizing the CZXT-013 job control, such as event and condition command, information notification and log file, log-out treatment and fault treatment, are discussed and their processing methods are also given.

At most, seven jobs can run simultaneously in the CZXT-013 system and one of these is an on-line job. Our system provides a functional and flexible job control language (abbreviated as JCL) suitable for scientific users. Both on-line and off-line controls are supported. Furthermore, in accordance with the characteristics of medium and large scientific users' computational procedures--testing, trial run and actual computation, we have made it possible to use both the on-line and off-line controls as well as alternating them for on-line jobs.

I. Job control language and operation manual

1. Job control language as system-user interface

The operating system extends the base machine to a functional, efficient and convenient virtual machine. Through the two languages provided by the system--the extended instruction set and the job control language--the user makes use of the various functions provided by the virtual machine. The extended instructions are used in the programs, and JCL is used in the operations manual or on the console. For users who use algorithmic language to write programs, the extended instructions are used implicitly.

Received 5 August 1980.

(The user uses directly statements in the algorithm language which are then implemented with the extended instructions by the compiler). Hence, JCL becomes the principal interface between the system and the user. JCL implements the system functions and it can be regarded as an explanation of the system. The design of the JCL, on the one hand, is confined by the design of the system functions and, on the other, it also affects the design of the system functions.

2. Design consideration of JCL

CZXT-013 is an operating system oriented toward scientific computations and, therefore, the JCL is also oriented toward scientific computations. In the design, we have taken into consideration the requirement for program testing and program execution control by scientific users as well as the existing keyboard instructions and the practical experience in using the testing and debugging methods in algorithmic languages, and have come up with a job control language that is functionally complete, flexible and easy to use.

When a user is preparing a job he has to use a variety of lang- 33
uages--algorithmic or assembly language, extended instruction set and JCL. For convenience, it is preferable that these languages be unified and standardized. In designing the JCL, we have paid attention to make the format of any statement as much the same as possible with that of the related statement of similar functions in the algorithmic language, e.g., the read/write commands. On the other hand, the extended instructions and the JCL with similar functions are unified through the "command" extended instructions. The user may store the command in the form of characters in a memory area,the first address of the area unit is used as the parameter for the command extended instruction. This extended instruction has identical functions as the corresponding command. Using this instruction, it is possible to write non-control type commands directly in a program.

The various commands in the JCL of our system, whether they be the operational commands used in writing the operational manual or the keyboard commands entered from the keyboard, or the commands appearing in the header of the paper tape, are all written in a unified command format to facilitate user job preparation.

The command format should be simple and easy to memorize. Such command implements a complete function for the user. Many commands take the form of macro commands. For example, the compile command really consists of a series of commands, such as the open command, read command, start command, etc. However, one should also consider the fact that it is not suitable to have too many parameters, or too complicated a structure. For the sake of making it intuitive and easy to remember, it may be desirable to break what may be unified as one command into several commands. For example, the assignment command and the set 0 (1) command may be unified into one read command. However, this will result in a large number of parameters as well as a lack of intuitiveness.

It is impossible to avoid extending or improving the JCL during the process of its usage; hence one must consider extendability in its design. Three ways are provided in our system for extending the JCL: 1. To extend JCL commands, it is only necessary to add one term to the verb table and add the corresponding command interpretation module to the command interpretation procedure; 2. Extension of service commands. A system service job is provided in our system to furnish some service items to the user. Each service command accomplishes one service item. The JCL function may be extended by increasing service commands. In this way, the main program of the system will not be affected. Only the system library files need to be extended by adding the service programs that interpret the extended commands; 3. Our system permits the user as well as each subsystem (i.e., each compiler system) to define, interpret and execute commands. A "user defined" command (the entrance to the user defined command interpreter program being given by the command parameter) is used to establish the connection between the system and the interpreter program for the user defined command. Thereafter, the user may then use at will the various commands so defined. (The user defined commands must be headed by the symbol ↑ to show distinction). This is the most flexible way for extending JCL functions.

3. A brief introduction to CZXT-013 JCL

The general format of JCL command is:

<label><verb><parameter><list><delimiter> [1)]

The label indicates which subsystem is involved or whether it is a user-defined command when the parameters are interpreted. The verb is the command operand and consists of two characters. Parameters vary with the command. Name addresses are allowed. There are 60 commands in the system. They may be classified according to their functions as follows:

(1) job set up category: To set up jobs, request resources, determine job category and provide command manual, etc.
(2) compiler category: To schedule various subsystems for compilation so that the commands of the executable target program may be obtained.
(3) file organization and management category: Commands such as set up file, open file, close file, delete file, modify tape file attributes, backup file, etc. There are 13 such commands.
(4) read/write category: Various input/output commands. In addition to the read, write and transmission commands, there are the console output, assignment (the content of assignment for some variable values is given in the command), set 0 or set 1 (set to 0 or 1 for some variable positions), clear (0 clear for whole group of memory units).
(5) testing and debugging category: Namely, commands used in testing programs such as tallied stop, timing (i.e., set program times), trace, backtrack, retain, restore copy, print status, etc.
(6) job run control catalog: commands for initiation and termination of job run, such as START, CONTINUE, MOUNT TAPE,

[1)] The symbol -- - - indicates that the item is optional.

PAUSE, LOG-OFF, DISMOUNT, etc. The condition commands used to process events in the offline control mode also belong to this category.

(7) operations manual control category: The operation manual may be regarded as a "program" that controls the running of the job. As for the general program, we have provided a set of commands such as transfer, return, reverse, etc. to alter the execution route of the manual and to organize suboperation manuals.

(8) Online/offline control modes switch category: For switching control mode of online jobs, consisting of commands such as transfer operation, return control, execute operation, continue execution, etc.

(9) service command: Commands used by the system in servicing the various service items set up by the job. One service item corresponds to one service command, e.g., compiling a program, transfering a file, etc. Once a service job is established, the operator needs only type in one service command to complete a service for the user.

(10) operator command: Commands used by the operator on the console to create, control and monitor system operation or to service offline user's command.

With the exception of categories (9) and (10) which are used by the operator, the commands in the various categories above are all used by users. Commands in categories (1)-(7) are the basic commands that should be provided by all common operating systems. We shall only introduce further some special commands of our system.

Taking into account the special characteristics of scientific users, we have provided in our system a complete set of debugging commands. The tallied stop command format is as follows:

```
FT S <position>; <frequency> or
FT Z <position>; <frequency>
```

They implement, respectively, numerical or index code, and will effect a tallied stop after going through a tally point n times. The trace command formats are:

```
ZZ  L  <trace>
ZZ  S  <trace point>
ZZ  A
```

to implement line tracing, value tracing and trace all respectively.

Trace point and trace range can also be specified. We have also provided keep copy and restore copy commands. The keep copy command format is:

BL <device number>, <logic record number>; <status table messages>.

The status retention table messages indicate what the user wants to keep. These messages may be given directly in the command or stored in the status table (when the contents to be retained are widely distributed) with the first address of the status table given in the command; the status table may also be dynamically constructed with the entry point to the status table construction program given in the command. The restore copy command format is similar. For scientific users with long computing time, these two commands are not only debugging tools, but also protection tools. Because the hardware and software stability in our country is still unsatisfactory, it is frequently necessary to use the keep copy command or extended instruction to retain copy during operation so that a suitable copy may be chosen, if necessary, to restore operation.

In a multioperation system, the functions to implement keeping and restoring copies are complicated but very necessary.

In the beginning of this paper, we mentioned that, for online jobs, one may use two methods for control and may switch between these at will which means that the online user may also use the operation manual. We have designed a set of flexible control switching methods and commands that maximize the advantages of the manual. When the

online user is furnished with the manual, the manual may enter the system first. An online job is established when the online user types in sign-on and request command on the console. Then when it is necessary to switch to offline operation, one needs only to enter a switch operation command ZC <operation code> to switch operation to the corresponding command in the manual. A return control command 338
FK is provided in the manual to switch from offline to online operation. When an online job is running under offline mode and a user enters any non-switch operation type command on the console, the system will switch control to online operation as it carries out the command. The system also provides the execute-operation command ZX <operation code> (to execute a command in the manual, similar to the execute instruction in the average machine) and the continue execution command → (to continue execute any command in the manual). With these commands the user may execute one by one the commands in the manual. Now the online operation mode will not change and it is not necessary to enter all the commands on the keyboard. Thus the operation is simplified, time is saved, and the flexibility of the online control is not affected.

Jobs are initiated by the initiation type commands. After the program is running, the system establishes the relations between the program and the user through occurred events (this is the manual for offline jobs). To process events and thereby control the running of the program, online users enter commands through the console and offline users through the condition commands set-up in the manual. The format of the condition commands is

RU <event name>; <operation code> or

RU <event name>; *BZ

This means: if event with <event name< appears, then execute the command with the <operation code> in the manual. *BZ means to carry out normal procedure when event occurs. At present, only one type of normal procedure is provided by the system--log-off, but it may be expanded to many types according to need. For <event name>, see the event table in part 2. An event name may also be *, indicating that any event that may occur are to be processed by the <operation> shown by the command or by *BZ.

There are two ways to arrange the position of a condition command in the manual: before the initiation type commands or after the initiation type commands. We adopt the former method because it facilitates the user to organize his manual and simplifies the implementation of the system. Condition commands may not appear at all indicating that normal procedures are to be used to process the corresponding events when they occur. If an event is to be processed differently after the next initiation, it is only necessary to write another condition command because according to the dynamic execution order of the manual, the condition command of the same event will take precedence if it is entered later. If no events require special processing after the next initiation, then only one RU*; *BZ needs to be written. Often several events may be processed in the same way. We may take advantage of the system-supported return command ZF <operation code> and reverse command FH to organize sub-operation manual. Thus, the set of condition commands and commands executed by the control manual as provided by the system is fairly complete. The user may organize simple operation manuals flexibly and conveniently.

4. Format and organization of the operation manual

The general format of the operation manual is:

<label section><command sequence><end section>

The label section is <manual name>: <comment> ∍, and the end section is only an end mark. In order to edit the manual with an existing editor, page and line marks may be freely inserted in the manual. 338

Ordinarily, the commands in the manual are interpreted one by one in sequence by the system. The path of execution is changed in the following situations:

(1) when transfer commands are encountered: the execution will be transferred to the appropriate command as for the transfer instructions in a program.

(2) when initiation commands are encountered: After the initiation type command is executed, the user's program starts running. The system waits for events to occur in the program. Once an event occurs, the system will transfer in accordance with the condition commands to the command with the corresponding operation code for execution or normal processing.

Hence, in a sense, condition commands and processing of various events are the nucleus of the operation manual. In Section 3, we have mentioned that condition commands must be written before the initiation command used in the current run. The positions of other commands in the manual are not restricted. They are organized according to the need of the user. In general, the organization of the operation manual is as follows:

(1) define logic device number--establish or open some files.
(2) read in program and data--some read command, compile command or mount tape command, etc.
(3) set some soft console values or switch values--with assignment command or set 0, set 1 commands. If necessary, some debugging measures may be set--with tallied stop command, timing, trace commands, etc.
(4) regulate how various effects are processed after the current initiation--some condition commands
(5) initiate user program
(6) process various events

For example: The program of a job A resides on magnetic tape, file name being PFILE while the data resides on paper tape with file name DFILE. Already from the entry command TRII=R; DFILE is already in the system. The job calculates coefficients and initial values first and starts the main calculation after the occurrence of the pause event READY. After every two minutes, results will be printed. In this run, it is also necessary to control the disc (i.e., the record on disc extended instruction is effective). The job requests a run time of half an hour and then logs off when the time is up. The manual of this job is as follows:

```
ASMS: EXAMPLE 1980/6 )             ) define logic device number, open the
      DK 12 = D; 112; PFILE; FX )  ) magnetic tape file on which the program
      DK 13 = P )                  ) resides and the disc files used for disc
           @ DU (12,0) PROGM )     ) recording in the program
             DU (11,0) DATA )      ) read in program and data

         RU ZT · READY; SMS1 )     ) after computing the coefficients and
                                   ) initial values go to SMS1

                 QD START )        ) start program running


                                   processing of pause READY:  set soft
           SMS1: Z1 KG; 0, 7 )     switch value to control disc recording
                 DK 14 = Y )       in the program during the current run;
                 JS 120 )          define logic device 14 as print file;
                 RU CS; SMS2 )     set program timer to 2 minutes; transfer
                 JJ )              to SMS2 when program time is up; continue
                                   start-up

             SMS2: GB 14 )         processing of events on program timer:
                   ZY SMS1 )       close print file 14, i.e., offline output
                                   result once, and then go to SMS1
```

II. Implementation of CZXT-013 job control

The control of job run is accomplished by the job control process. Job control process is the first process of the job family. Once the high level scheduler schedules a job to run, the job control process is then established and initiated by the high level scheduler program. Then the various processes of the job family are established and initiated in an outward direction according to the level structure of the job family by the job control process. When the job finishes running, the various processes of the job family are stopped and deleted one by one in an inward direction by the job control process. Thus the job control process establishes, controls, and deletes the various processes of the job family.

The online job control process must process the reception and implementation of keyboard commands while the offline job control process must control and implement the operation manual (another command interpretation process is provided in our system to help process command interpretation). Some extended instructions of a controlling nature are also processed by this process. During the run the job continuously requests system services and produces various events. The job control process must report the service status and event production as well as process the events, particularly the wrap up processing during log-off and error or fault events. We shall only introduce the processing of a few key problems in the implementation of job control.

1. Events and condition commands

It has been mentioned above that the production of events is the major means with which the system establishes a program and communicates with the user after the job starts to run and that the condition commands are the major means with which the user controls the running of the job offline.

The initiation of events is related to the functions provided by the hardware and software. It also should enable the user to control the running of the job flexibly with these events. According to their mode of production, events may be classified into four categories:

(1) events produced by hardware--e.g., events caused by error interrupt due to main frame error and program interrupt due to program error, such as computational overflow, illegal operation, etc.

(2) events caused by hardware/software combination--e.g., program timer up, tallied stop, etc.

(3) events produced by software--e.g., various syntax errors of commands and extended instructions. These events may be augmented and may be set up in a flexible and diversified way. In particular, subsystem pause extended instruction is supported in our system, based on which many types of events may be produced in conjunction with various subsystems, such as subscripted variable address out of bound, etc.

(4) events set up by the user to communicate with the program, e.g., pause event, wait (for instruction) event. Pause event is produced by the pause extended instruction. Its parameter (halt number) may be a character string of no more than six characters or an octet number of no more than six digits. Different pause events are produced by pause instructions with different half numbers. The pause instruction of the subsystem mentioned above produces pause event of the subsystem. It is the same as the pause extended instruction except that it is exclusively used by the subsystem. The pause extended instruction provides the user with the capacity to install event production at will so as to establish communication with the program. The user may tactfully install pause instructions at various control points in the program to produce pause events. With various condition commands installed after the pause instructions, one can then flexibly control the running of the job.

A "wait" extended instruction is also supported by our system to produce a wait event. For offline users,its function is similar to the pause event, but for online users, it may be used to communicate with the program. This extended irstruction carries with it a parameter pointing to the leading address and length of the storage area into which will be stored the content input from the keyboard by the user. The program will pause when it comes to a wait instruction and wait for the user to input a message from the keyboard. The input message will then be stored in the area indicated by the parameter. The system will then continue to run and may make use of this message.

The event table provided by CZXT-013 is as follows:

SY	computation overflow
HY	H register overflow
DZ	illegal address
CZ	illegal operation
CJ	main frame error
YF	syntax error for commands or extended instructions

XX	external message error	
IO	external device error	
DS	job timer up	
CS	program timer up	
FT	tallied stop	
DD	wait instruction	
ZT	pause	different halt number for different events
TT	subsystem pause	

Three tables are constructed by the system--the event table, the pause event table and the code-address cross reference table, to implement job control by condition commands. Since pauses with different halt numbers are considered as different events, the number of pause events will vary with the job. Hence a pause event table is set up separately. During scanning of the operation manual, the system tallies the number of pause events before requesting for storage space, building tables and establishing linkage with the event table, so as to save storage space overhead. Process messages for events indicated by corresponding condition commands are stored in the event table and the pause table--such as various operation codes or normal processing. Their initial values are all normal processing messages which means that in the absence of a condition command in the operation manual, the corresponding event will be processed according to the normal process when it occurs. When a condition command is encountered as the system processes the commands in the manual one after another, the event process shown by the condition command will be entered in the corresponding entry in the event table or pause table. Thus, the order of dynamic processing of the condition commands of the same event will be last in first out in accordance with the operation manual.

In the code address cross reference table are stored the relations of correspondence between the operation code in the operation manual and the position of this code in the manual (unit address and character address). To save memory space, we process it in the same way as with the event table, namely, that the number of codes in the manual is tallied first, then the memory space is requested and the table is constructed as the manual is scanned again.

In processing the condition commands, the position messages of the operation code in the manual are entered in the corresponding entries of the event table or pause table through checking into the code address cross-reference table. Once an event occurs, the job control process will search the event table or the pause table to obtain the process message of this event. If it is an operation code, then the corresponding message is used to change the pointer in reading the manual and fetch the corresponding command for processing. If it is to be the normal processing, then the entry point to the processing program will be furnished by the table and transfer will be made to the entry point for processing.

2. <u>Message communication and monitor file</u> 341

Although CZXT-013 JCL is not an interactive job control language, much effort is made to provide the user with enough message communication and run record. For each job, the system establishes a monitor file to record the occurrence of various events and the corresponding process during the run. The major content of the monitor file is: operation manual; command dynamic processing status (command and command reply); control type extended instruction and command extended instruction processing status; communication message of all events; console output message; trace command message; print status message, job resource utilization status, etc.

The system provides in the job family a special write monitor file program to form the monitor file. This is to improve parallelism, to guarantee as much as possible the timely output of message communication and to make it easy to proceed with special processing under error conditions.

To provide enough communication as well as a unified report format for the user, our system adopts the implementation method of having the processes of a job family report upward from lower to higher levels, finally concentrating in the job control process for processing. Communication and processing are done in a unified manner by the control process. For a large scale operating system to have a detailed as well as easily understandable message report requires a large work load. How to strike a happy medium is a problem that must be solved for a practical operating system.

3. Log-off processing and error processing

The sign of completion of a job is log-off. The log-off of a job may be classified into the two categories of normal log-off and abnormal log-off. Log-off arranged by the user--at end of a job run or termination of run due to certain event--is normal. Job forced to terminate and to log-off due to system error (hardware or software) is abnormal. Sometimes, in order to balance the system load, increase system throughput or prevent deadlock, the operator needs to force a job to log-off (through operator command). This too is abnormal log-off.

When a job logs off, the system needs to do a great deal of wrap up work, e.g., waiting for the completion of services requested by the user; closing all files; rewriting magnetic tape header file, retrieving all the resources used by the user and finally, stopping and deleting the whole job family, etc. In order to accumulate system property data, provide the user with resource utilization status for the job and facilitate accounting, the system keeps certain statistics for some data during the running of the job. A resource utilization status table is output for the user. The system also keeps a record.

Abnormal log-off is in fact error processing. It is complicated but unavoidable and yet often neglected in the design. In our situation of unstable hardware and lack of experience in software, error processing is all the more important and difficult. Error processing 342
directly affects the security and reliability of the system. When it is done well, one can prevent the loss of resources and the prevention of deadlock. A good structural design will diminish the complexity of error processing. On the other hand, error processing will also affect system design. It may increase the complexity and work load of the

system. A problem that requires careful consideration at the early stages of design is the balance of give and take.

Limited by the lack of hardware disk and tape, their low speed and instabiility, the lack of support for a powerful file system as well as experience and manpower, we do not have various types of error recovery in our system. Thd basic principle of job family error processing in our system is: to limit as much as possible the extent of loss and the effect of error; to retrieve all the resources used by the job and to avoid system deadlock.

When error appears in a job family process, the system treats it as abnormal log-off. We divide abnormal log-off into three levels:
(1) job log-off when a program error or first error interrupt or program interrupt occurs in the job family system process (non-job control process). At this time the job control process can still run but it no longer can delegate work to subprocesses. For the user, it is a destructive log-off but the resources used by the job can all be retrieved and there will be no effect on the normal running of other jobs or the system.
(2) error interrupt program interrupt appearing in the job control process or the second error occurring in the other system processes in the job family or the operator's order forcing the cancellation of the job. The job control process has already been forced to stop and may not be able to run normally. A higher level scheduling process forces the initiation of the job control process to complete the log-off of the job. If other errors should occur in this process, then we have the third level.
(3) a second error occurs in the job control process. Now the job control process can no longer run normally. A higher level process will complete the log-off procedures for the job.

As far as the extent of error is concerned, these three levels are all limited to the current job itself. However, the measures taken are different in order to retrieve all the resources and to avoid deadlock.

The complexity of abnormal log-off processing due to errors is principally due to the fact that an error may occur in any process at any moment. After the occurrence of the error, the process can no longer run normally. In general, it can only stop and wait for cancellation. The principal problem that follows is:
(1) mismatch of matching communication relation under normal conditions. To solve this problem, we have to provide first the means of checking communication mismatch. Secondly, once a mismatch is discovered, it is generally not possible to rematch through the process in which the error has occurred. We must take steps to use other processes to do this.
(2) the wrapup process during log-off cannot proceed normally. The resources that can be completely retrieved under normal conditions cannot be so retrieved, especially those resources used implicitly by the system--such as I/O buffer, message buffer, etc. For these problems, appropriate measures have been taken in our system (see the article "Some problems in the implementation of CZXT-013" in this issue).

III. CONCLUSION

Job control language is the actualization and description of the system functions. Its design is closely related to user requirement. As the operating systems increase in number nationally, the unification and standardization of JCL, the unification of JCL and program design language and the research on machine-independent JCL must all be discussed. We have encountered a number of important problems in the engineering practice of implementing job control, such as highly efficient and secure process communication tools, error recovery and error check, testing and prevention of deadlock system, security and reliability, system protection, etc. These should all be studied further.

THE JOB MANAGEMENT IN CZXT-013

Jin Qu-jie, Zhen Zeng-Zhen

(The 9th Research Institute of Beijing)

Zhang, Cong-Min

(Institute of Computing Technology, Academia Sinica)

ABSTRACT

CZXT-013 is a batch processing operating system which also facilitates on-line operations. The system can accommodate three different types of jobs to operate simultaneously. This paper introduced the job scheduling scheme of the CEXT-013 system which is realized in three levels. This paper also introduced the states of the job in different stages from entering to leaving the system; the state transitions; the preparations for the job, and the handling procedures of jobs under normal and abnormal departures.

I. INTRODUCTION

In an operating system, the portion involving the establishment, scheduling, operating control and departure of jobs is usually called "job management". With the exception of operating control of the jobs which will be discussed by another paper, all other problems concerning the job management in the CZXT-013 system is going to be discussed in this paper.

A job in the CZXT-013 system is defined as an independent operation request made by a user to the system. Each job has its name, and the names in the system cannot be repeated at the same time.

The CZXT-013 system permits three types of jobs: off-line operating job, on-line operating job and system service job (from here on they are called on-line job, off-line job and service job).

(1) Off-line job. The operational control of the job is completely controlled by the operating manual of the job. The input of key punched messages (including program, data and operating manual) of the job and printing of the output message must go through the input/output buffer (i.e., the input and output are carried out by the so-called pseudo off-line subsystem).

(2) On-line job. The operation of the jobs is controlled by the user from the keys of the user control desk (control desk for short). The key-punched message and the printed message not only can go through the input/output buffer but also can directly input and output using external on-line devices.

(3) *Service job.* *This type of job* is to finish some service oriented jobs for the system. For example, to transfer information stored in an external medium to another external storage medium, the treatment of source program with compilation but no computation, etc., are service jobs. The job operation is carried out according to the standard program of the system. The processing of key-punched information and printed information is the same as that of on-line jobs; it is possible to use either input/output buffer or direct on-line input/output.

This paper was received on August 5, 1980.

The CZXT-013 at most will permit the simultaneous operation of 345
one service job, one on-line job and five off-line jobs in the system. In addition, it allows the reserving of several off-line and on-line jobs waiting for operation in the back-up job key (but cannot back-up service jobs waiting to be run).

The job scheduling of the CZXT-013 system is realized in three levels, i.e., high level scheduling, medium level scheduling and low level scheduling. High level scheduling involves the selection of part of the jobs already assigned to the system to establish the virtual space, to assign special input/output devices and to set up the initial progress of the entire family of programs for the jobs--

manual. We have designed a set of flexible control switching methods and commands that maximize the advantages of the manual. When the

job control process. Medium level scheduling is responsible for the actual assignment of real storage page for the job already chosen by the high level scheduling. Low level scheduling involves the assignment of processor time to progresses (including the user progress and system progress belonging to a job) of a job that already has an actual storage page. The departure of jobs is divided into normal and abnormal departures. Certain necessary procedures are taken to process abnormal departures in the CZXT-013 system to prevent the system from being incapacitated and prevent loss of information. With the exception of the details of medium level scheduling which will be covered in a separate paper, other problems are going to be discussed in this paper.

II. STATES OF JOBS AND STATE TRANSITIONS

No matter what type of job is involved from entering to departing the system, it is necessary to go through various stages in the system. In order to describe the various stages, the CZXT-013 system defines the following states for all the jobs:

(1) Submission state: This is a state used to describe the process of jobs from entering the system to completion. When a job enters the system, the system sets up a job control block (JCB) for it. The system also records the information provided and produced by the job in the JCB.

(2) Back-up state: This is the state when the entering of the job to the system is finished, JCB has been established and it is awaiting the selection of high level scheduling.

(3) Waiting state: The virtual space necessary for the entire operation has been established and the special input/output devices have been dispatched. But the real storage page space has not been assigned, therefore, the processor time cannot be requested yet.

Under the two conditions described below, jobs will enter waiting states: one is a job in back-up state which is chosen by high level scheduling to enter from the back-up state to the waiting state and the other is a job in a running state which loses its real storage page by medium level scheduling so that it is forced to return to the waiting state from the running state.

(4) Running state: It is a job state which has been selected by medium level scheduling. At this time, it has already assigned a certain number of real storage pages. From a broad view, it can be considered that the job is running. From a microscopic point of view, it may be running on the CPU or it may be waiting to run on the CPU.

(5) Completion state: It is the state during which a job finishes running and completely departs from the system. At this time, the system is taking care of the aftermath (including retrieving the information resources).

Figure 1 shows the possible transition between various job states. One point must be explained individually: it is not necessary for every job to stay in the back-up state for a period of time. As long as the job satisfies the scheduling conditions at that time, it can go to the waiting state through a job transition directly without staying over in the back-up state.

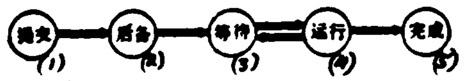


Figure 1. States of job and transitions of state
Key: 1--submission; 2--back-up; 3--waiting; 4--running; 5--completion

III. ESTABLISHMENT OF JOBS

Under the control of CZXT-013, jobs are entered into the system through two routes: one is called the pre-entering of jobs and the other is the direct entering of jobs. All the jobs can be pre-entered into the system but only on-line jobs and service jobs can be entered directly. Off-line jobs cannot be entered into the system directly.

1. Pre-entering of jobs

Before running a job, the information required to run the job and the whole or part of the key-punched information needed to run the job are sent into the system ahead of time which is called the pre-entering of jobs. In order tc realize pre-entering of jobs, the CZXT-013 system is designed to allow the appearance of four types of preceding commands on the leading edge of the paper tape of the job. The four preceding commands are: escape entering command, request command, job type command and request for system service command. Among them, the escape entering command is applicable to all the jobs. The request command and the job type command are used for on-line and off-line jobs,but not for service jobs. The request for service command is specially used for a service job and cannot be used for other jobs.

(1) The escape entering command: The escape entering command has the following format:

#TR < logic equipment name > = <paper tape name>(<information section name> ,...., <information section name>); <paper tape name> (<information section name>,..., <information section name>);....<paper tape name> (<information section name>, ...<information section name>);[1)]

The escape entering command is used for defined document whose < logic equipment symbol > is paper tape. It notifies the system to send the indicated paper tape information to the input of the system and the correspond these input images with the < lɔgic equipment symbol > in the command. When the job is running, it is also possible to read this paper tape information on the logic equipment.

(2) Request command: A request command is used to request resources of the system for a job. Every pre-entered off-line job and on-line job must have a request command.

1) The dotted lines in the command format indicate that that portion is a selected term

(3) Job type command: The job type command is used to provide the relevant information of the job to the system on the type of job (off-line or on-line); the position of the storage space of the operating manual, and the symbol of the beginning of execution of the operating manual. Every off-line job and pre-entered on-line job must have a job type command in the preceding command series.

The format of job type command is:

$*LX\begin{Bmatrix}L\\T\end{Bmatrix}$; <operating manual information>; <operating symbol>;

where "L" indicates that this job is an on-line job, while "T" represents an off-line job. <operating manual information> can be a magnetic tape document name or a logic equipment symbol[2)] of a defined escape entering paper tape document, or a standard operating manual symbol. For on-line jobs, it can even be blank. When the <operating symbol> is blank, execution begins with the first order of the operating manual. When the <operating manual information> is the standard operating manual or a blank, <operating symbol> must be blank.

(4) Request for system service command: The request for system service command is used when the system service jobs request the system to offer a certain system service and to provide the necessary parametric information required for the service. At the present moment, the pre-entered service job is limited to editing (only editing, no execution). Other service rquests will not be processed (but can be processed by directly entering the system through the operator at the control desk).

The pre-entered information of each pre-entered job is usually 34
composed of two parts: the preceding command series and the job running information. The former is formed by several preceding commands and the latter consists of the program, the data and the key punched information of the operating manual. All this job running information is punched on one or several paper tapes. Each paper tape

2) At this time, there must be an escape-entering command in the preceding command series which defines the corresponding storage equipment after the key punched information has entered the system.

job control block zipper indicator					
job name					
channel no. when the job is running	job control type, T/off-line, L/on-line F/service	job running character-istics type	requested service item of system service jobs	no. of doc-uments escape entering	no. of magnetic tapes on request
• time at which the job enters the system					
the latest time at which the job begins to run					
actual time at which the job starts to run					
time at which the job finishes running					
}					
} information on the operating manual relevant to storage					
}					
symbol of the geginning of execution according to the operating manual					
virtual storage space requested by the job (using pages as the unit)					
actual storage space of the job required by the job (using pages as the unit)					
running time required by the job (using minutes as the unit)					
each requested magnetic tape unit has a corresponding unit which gives the reel number of the magnetic tape used on the magnetic tape recorder) not to exceed 4 units) each unit not to) exceed 5 reels					
each entering document has a corresponding unit which records) the first zone corresponding to the document) not to) exceed 20					
total no. of zones occupied by the positions and the corres-) ponding logic equipment symbol)					
job control block termination symbol unit					

Figure 2. Job control block (JCB)

can be divided into several information segments; or it is not divided at all, which is indicated by the paper tape label at the beginning of the paper tape. If the paper tape is divided into information segments, then each information segment must be labeled with different information segment names. Each paper tape must be punched with a lead including <job name>, <paper tape name> and <paper tape type characteristic information>. In all the paper tapes of a pre-entered job, only the leading part of one paper tape must contain a <preceding command series>. Under the most special condition, all the punched input information of the pre-entered job is only the leading part of the paper tape containing a <preceding command series> (i.e., there is no key punched input information needed to be read in the running of the job).

2. Direct entering of the jobs

The other route to enter a job into the system is direct entering. During on-line jobs, the users can use the control desk to enter commands such as "sign-in" and "request" and in system service jobs, the operators at the control desk can type in "request for system service" command to enter the system. In these commands, they contain all the information such as job name, resource request, requested service item, etc. needed to establish the corresponding job in the system. Off-line jobs cannot be entered into the system directly. When the directly entered job is running, the requested key punched input information is sent in from the on-line input in operation.

3. The establishment of jobs

When the job is pre-entered into the system, the duty of the system is to process the preceding command series. During the course of processing the preceding command series, it also establishes the escape entering paper tape document (if there is an escape entering command) for the job and fills out the contents of the job control block for the job. Each entering command causes the establishment of an entering paper tape document. The image of all the key punched information belonging to the document is placed on the input device (magnetic disc) according to a serial document structure. The position of the first zone and the number of zones corresponding to the document as well as the corresponding logic equipment symbol are recorded in the job control block. The system requires that the number of entering command for each pre-entered job cannot exceed 20. The information provided by other preceding commands is also recorded in the job control block.

After the processing of the preceding commands is complete, the corresponding control blocks of each pre-entered job are keyed into the off-line back-up job key or on-line back-up job key to await for the selection of high level scheduling. At this time, the submission state is shifted to the back-up state. For system service jobs, the system is not equipped with a back-up job key. Therefore, at any time, the system only allows one system service job.

When the job is entered into the system directly, the system establishes a job control block after receiving the job establishment command (e.g., "sign-in" or "request" or "request for system service"). It also enters the relevant information given in the commands and then the job is shifted to the back-up state.

It must be pointed out here that in order to schedule the on-line jobs and system service jobs entered to the system directly with priority, their control blocks in the back-up states are not placed together with the back-up jobs pre-entered into the system. They are specially placed individually at two special keys. In addition, at any time, each key has only one job as the maximum.

The detail of the job control block (JCB) is shown in Figure 2.

IV. SCHEDULING OF JOBS

The scheduling of jobs not only involves the distribution of the processor but also is related to the distribution of the resources of the entire system. The CZXT-013 system uses high level, medium level and low level scheduling, to realize the scheduling of the entire system.

The CZXT-013 is primarily a batch-processing oriented operating system. Therefore, in the design of the method of scheduling, the main problem considered is the increase of system efficiency, especially the utilization efficiency of the CPU. For that, the system adopts the following scheduling principles:

(1) To divide the jobs into three types according to the running characteristics: computation-oriented, input/output oriented and the usual type. High level scheduling, when making the selection from the back-up jobs, will make an attempt to balance between computation and input/output.

(2) When each off-line job is entered into the system, the system assigns a "latest time to begin running the job". The sequence of this time actually reflects the priority from the "back-up state" to the "waiting state" or the duration allowed to stay in the back-up state. The "latest time to begin running the job" is obtained by adding an increment to the "time at which the job enters the system". When calculating the increment, we have to consider the amount of special equipment required by the job, the size of storage memory and the duration of the running time. Small jobs will get the priority. Howver, large jobs are also assured to be scheduled after spending a certain amount of time in the back-up state.

(3) The CZXT-013 is a virtual storage operating system. In order to assure the efficiency of the system, the use of real storage is a key. For that, in the scheduling method a medium level scheduling is established to monitor and adjust the use of real page with respect to each running job to avoid the occurrence of "bumpiness" which causes drastic reduction in system efficiency.

(4) When jobs are running, to the extent possible, the system assures that a job is in "real job state",which means that during running it will not produce a job with "page fault" to ensure that when

other running jobs have page faults the CPU is still utilized. 349

(5) The operator can ensure the rational utilization of the resources of the system by changing the off-line job channel number allowable for running by the system (by establishing an operator's command to realize this function).

(6) Using a ready line-up station with various priorities. When the processing progresses into the ready line-up station, it enters different lines according to the characteristics and the reasons for the blockade. The system progress always enters the highest priority ready line-up. The user progress proceeds based on the various causes releasing the CPU such as (1) page fault (2) on-line input/output (3) transfer of document (4) expiration of time piece, etc., to determine which line-up it is going to enter. When the ready progress gets into the CPU, the system progress assures to give a long enough time piece. For a user progress, the high priority ones are given shorter time pieces while low priority ones are given longer time pieces. The advantage of adopting this scheduling principle is to reduce the number of CPU carving to reduce the "expenses" of the system.

V. THE SCHEDULING OF ON-LINE JOBS AND SYSTEM SERVICE JOBS

In order to ensure that on-line jobs can be scheduled at any time, a certain amount of on-line job resources is reserved. These resources are: a 260 K storage memory (including virtual storage and disc storage), two magnetic tape drives and a printer. The request for on-line job resources will be immediately scheduled after typing in the "sign-in" and "request" commands within these limits regardless whether the running characteristic type of the job is proper. If the request for resources exceeds the limit and the present system cannot be satisfied, then it is necessary to wait (i.e., staying at the back-up state). At this time, the system temporarily suspends the scheduling of the off-line job to satisfy the resources needed for the on-line job with priority.

The system does not reserve any resource for system service jobs. However, when the resources of system service jobs cannot be satisfied, similar to the case of on-line jobs, priority is given to satisfy it first.

VI. PREPARATION OF RUNNING THE JOBS

When a back-up job is chosen by the high level scheduling, the high level scheduling must make various preparations for its running. These mainly include:

(1) to distribute a running channel number for the job;

(2) to assign special input/output devices for the job. Here it mainly indicates magnetic disc drives;

(3) to establish the virtual space of the job. This includes the dispatching of back-up storage device (magnetic disc) based on the requested storage capacity and the establishment of page table. Because the page table zone of each job channel has fixed storage position for the 013 machine, in order to raise the utilization of the real storage page, these real pages can be occupied by the jobs in other channels before the page table is established. Hence, at this time it is necessary to determine whether the page table zone of the job channel is blank or not. If it is occupied, then the occupier will be ordered to liberate this page space. Afterwards, page table can be established;

(4) to establish "user document index" for the job. The CZXT-013 requires that each job channel can use 32 pieces of logic equipment. Its octal symbols are 0-37. The input and output as well as the access and storage of document during the running of the job can be opened to (i.e., defined to) a certain logic device using the "open" command ahead of time and then carrying out reading and writing from that logic device. The 0, 1, 35, 36 and 37 logic devices among the 32 logic devices are opened consistently by high level scheduling

which is also defined by the system. They are defined separately as controlling the image printing, system printing and monitoring document, etc. 350

When pre-entered jobs enter the system, the input document defined by the user with the relevant information orginally recorded in the JCB will be moved to the "user document index";

(5) a running priority number is calculated based on the operating type of the job and the amount of resources requested. Medium level scheduling based on this priority number chooses the running of the job from the jobs in the waiting state;

(6) to establish and initiate the initial progress of the job channel--job control progress. Situation concerning the job control progress is shown in a separate paper.

VII. DEPARTURE OF JOBS

Under normal conditions when the operation is finished, the job control progress after completing its finishing jobs issues a message to high level scheduling to cancel this job. The job departure work is the reverse of the preparation work before the running of the job which will not be repeated here. The emphasis here is to discuss the points to be aware of under the abnormal departure of a job.

When the job progressing process has a program breakdown or when the main unit breaks down and can no longer normally coordinatively execute to the end, the abnormal job departing problem occurs. Because the timing of the job progressing process breakdown cannot be predicted; therefore, the abnormal departure of jobs cannot be predicted either. Thus, the following situations may occur: the system may have sent a request for some message to the job progressing process. But it does not have time to process and to issue a return message. If, at this time, this job and this family of jobs are cancelled, then the system family will wait indefinitely for the return message. For this, the system specially sets up a "closed message" type. The high level scheduling is issuing this type of message to the awaiting system group to inform them that the progress has been cancelled in order to maintain the balance of messages among them.

VIII CONCLUSIONS

The structure of the job management of the CZXT-013 system is basically clear. Therefore, the reliability is also high. The problem now is the efficiency of the system. As for the design of the scheduling of computation, although we have considered various factors, there is no feeling about which point to deal in the actual system (hardware characteristics and usage characteristics). Therefore, in future running, we should accumulate the relevant system characteristic testing data for analysis in order to improve the present method.

MANAGEMENT OF VIRTUAL STORAGE ON CZXT-013

352

Fan Ben-Kui

(Institute of Computing Technology, Academia Sinica)

ABSTRACT

CZXT-013 provides a page multiple virtual storage system. This paper introduced the management technique of virtual storage from the software point of view. It described how virtual storage is realized from the auxiliary storage management, mass storage management and page management (page fault treatment, page replacement rule, etc.). It then discussed the efficiency of the virtual storage and analyzed the effect of several prime factors on the efficiency of the virtual storage.

I. INTRODUCTION

CZXT-013 provides a page multiple virtual storage system to the users with a maximum of seven virtual spaces.

In order to realize the virtual storage system, a dynamic address transfer mechanism is installed in the command control part of the 013 machine. It primarily consists of: a page table reading and writing circuit, high speed page table storage and comparison circuit (i.e., associative storage), page protection circuit and the replacement counter and writing circuit (small table) which reflect the utilization of main storage situation, etc. The function of this dynamic address transfer mechanism can be summarized as follows:

(1) based on the page address transfer table provided by the system, it transfers the virtual address given by the job into the real address pointing towards the main storage. When the transfer is unsuccessful, a page fault is created which is processed by the operating system;

(2) together with the operating system, it provides the proper protection to storage.

(3) through the use of a small table, it provides the sequence of main storage page utilization to the operating system.

The working principle of the address transfer mechanism is shown in Figure 1, where XY is the virtual page number, SY is the real page number, JS_{tt} is the value of the replacement counter, G_a is the corrected indicator position, D_o is the dynamic indicator position and Z is the address in a page.

The virtual storage system CZXT-013 is realized by using a main storage with capacity of 130,000 characters in connection with three movable arm magnetic disc storage,each with a 850,000 character capacity,on the basis of the dynamic address transfer mechanism.

All the jobs must use the virtual space whose maximum is 2^{20} characters. The length of the virtual address is 21 digits. The lower 9 digits are address within a page and the higher 12 digits are virtual page numbers. The highest digit is always 0. The programmer believes that this one-dimensional virtual space storage medium is homogeneous and continuous.

According to the usage requirements and the presently available hardware conditions, the virtual storage system we adopted has the following characteristics:

(1) it provides a number of mutually "insulated" page virtual space;

(2) the virtual space of a job is a combination of real and virtual spaces. The virtual space is used to obtain a high capacity and the real part is used to ensure high CPU efficiency;

This paper was received on August 5, 1980

(3) the distribution of job information in auxiliary storage uses a combined longitudinal and lateral method. The longitudinal support can use different magnetic discs to support various jobs to reduce

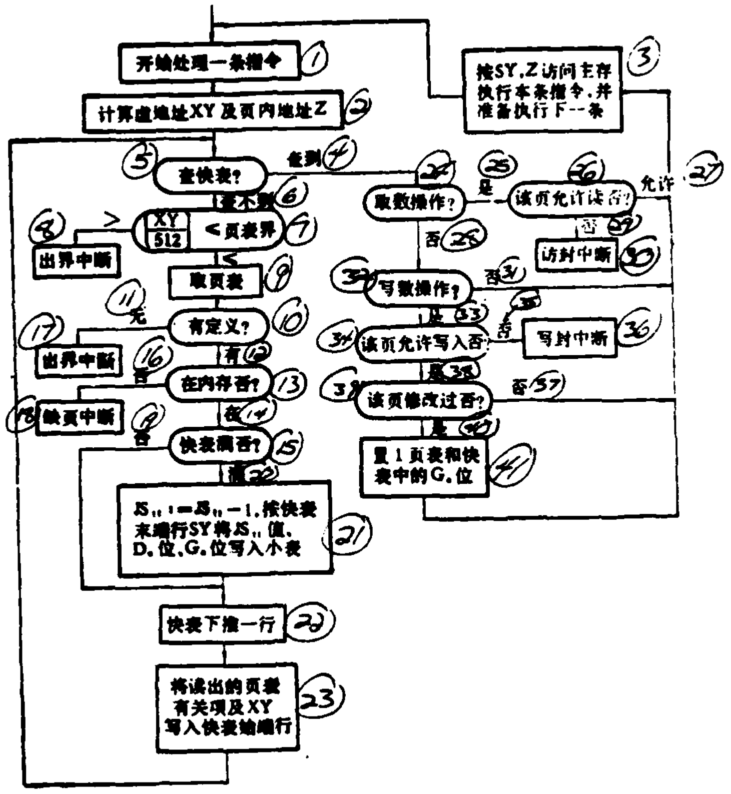


Figure 1. Schematic diagram of the working principle of the address transfer mechanism

1—begin to process a command; 2—compute the virtual address XY and page address Z; 3—visit main storage according to SY and Z to execute the order and ready to prepare for execution for the next one; 4—found; 5—check the quick table?; 6—not found; 7— $\left[\frac{XY}{512}\right] \leqslant$ page table boundary; 8—out of boundary interruption; 9—take page table; 10—defined?; 11—no; 12—yes; 13—in the internal storage?; 14—yes; 15—is the quick table full?; 16—no; 17—out of boundary interruption; 18—page fault interruption; 19—no; 20—full; 21—when $JS_{tt+1} = JS_{tt} - 1$, write the values of JS_{tt} according to the last row of SY in the quick table, the D_o position and the G_o position into the small table; 22—push to the next row in the quick table; 23— write the relevant terms of the page table read and XY into the initial row of the quick table; 24—get data to operate?; 25—yes; 26—is this page allowed to be read?; 27—yes; 28—no; 29—no; 30—visit interrupted; 31—no; 32—data writing operation?; 33—yes; 34—is this page allowed to be written into?; 35—no; 36—writing interrupted; 37—no; 38—yes; 39—has this page been corrected? 40—yes; 41—place (page table and the G_a position in the quick table

mutual interference between jobs and the lateral distribution can increase the capability of the magnetic disc to provide page space;

(4) the sch-duling of storage and the scheduling of the processor

are closely matched to keep the system running at high efficiency.

This paper primarily introduces how the virtual storage system is realized from the point of view of storage management. Finally, initial analysis is made with regard to the efficiency of the virtual storage system.

II. THE MANAGEMENT OF THE AUXILIARY STORAGE

The main mission of the management of the auxiliary storage system (auxiliary storage for short from this point on) is to distribute and release the auxiliary storage. This part of the program is a component of high level scheduling. When the high level scheduling selects a job from the back-up state to enter the waiting state, or when an on-line job enters the system, auxiliary storage is distributed according to the requested virtual space size. When the job is finished with normal or abnormal departures, the occupied auxiliary storage is taken back.

The 013 machine currently has four simultaneous arm reaching and positioning movable arm magnetic discs. Three of them are used as the auxiliary storage for the virtual storage system and the other one is used as the input/output well. The technical standard of the presently available magnetic disc is relatively low. The average waiting time is 750 ms out of which the vibration stabilization time is 250 ms. Therefore, the movement of the magnetic head arm requires a long period of time. Despite the optimization treatment with regard to the reading and writing of the magnetic disc, the capability of the magnetic disc to provide page space can still not catch up with the demand of the processor on page space,which means that there is an apparent mismatch between the supply and demand of page space. Because the supply demand mismatch problem of page space is an important factor affecting the efficiency of the storage system, the management of auxiliary storage becomes more important. Based on the characteristics of the magnetic disc of the present 013 machine, in order to relax the contradiction between supply and demand of page and to sufficiently utilize the parallelism of the magnetic discs and to reduce the arm movement time and distance to the extent possible, we adopted a combined longitudinal and transverse static distribution method.

The so-called static distribution is to distribute the entire requested virtual storage capacity of the job at one time and actually locate it in the real position without considering the contraction and expansion of the virtual space. The so-called longitudinal and transverse combination is to use longitudinal and transverse distribution methods according to the size of the virtual space.

(1) Transverse distribution method

Because the capacity of each zone (cylindrical surface) of the present magnetic disc in the 013 machine is 32 K characters and the reading/writing magnetic head arm cannot be shifted in the zone; therefore, for those jobs with a virtual space smaller or equal to 96 K characters we use the transverse distribution method based on alternating units using the zone as the unit and the segment as the count (each segment has 512 characters). The jobs using transverse distribution are called lateral jobs. This corresponds to providing a fixed magnetic disc to the lateral job so that the page can be quickly given. However, because a lateral job must simultaneously occupy several magnetic head arms (the number to occupy depends on the virtual storage of the job), it tends to repel other virtual jobs. Therefore, the number of lateral jobs in the waiting and running states cannot be too large. This system requires that the number is not to exceed three. The matching of the lateral job with the real state job whose virtual space is entirely in the internal storage can assure the highly efficient running of the CPU.

(2) The longitudinal distribution method

For all the jobs with a virtual space exceeding 96 K characters, we adopted the longitudinal distribution method,which uses the zone as the unit, the segment as the count and the information is concentrated on one disc. Furthermore, the information on one disc is stored in a concentrated manner to the extent possible. The actual method is to first determine the unit number which always begins with a disc with the most number of free segments. If one unit is not enough, the unit with the next high free segment number is assigned until the virtual storage requirement is completely satisfied. After the unit number is determined, the block is defined. The principle to define a block is to assign the smallest free block capable of containing the entire virtual storage. If all the free blocks in the same unit cannot accommodate it, then the largest free block is first distributed and then followed by the second largest and so on until the requirement is satisfied or until all the free blocks in the same unit are completely distributed. The so-called free block is a block formed by neighboring free zones. The jobs using the longitudinal distribution method are called longitudinal jobs. The advantage of the longitudinal distribution method is that to the extent possible different discs are used to support various longitudinal jobs to reduce the interference between various jobs to facilitate the running of the multiple path program. The concentration of information storage can also improve the capability of the disc to provide page space.

In addition, the management of auxiliary storage can eliminate the capability of the breakdown segment on the disc.

III. THE MANAGEMENT OF THE MAIN STORAGE

The primary mission of the main storage system (main storage for short from here onward) is to distribute the real storage page for the jobs chosen by the high level scheduling. Simultaneously, it dynamically adjusts the load of the main storage according to the actual demand to provide conditions for improving the running efficiency of the system. In this system,the medium level scheduling is in charge of the management of the main storage. The running of the medium level scheduling progress is carried out at fixed times.

When a job is chosen by the high level scheduling, it only provides auxiliary storage for the job in order to establish the virtual space and to distribute the special external equipment (e.g., a magnetic tape disc). However, at this time the real storage page is not dispatched and the state of the job is the waiting state. Only when after the medium level scheduling dispatches a certain amount of real storage page for a job in its waiting state, the job can obtain CPU. At this time, the state at which the job is located is the running state.

35

When the main storage load is too light, if there is a job in the waiting state, then the proper amount of real storage page is distributed to the job so that it can transfer from the waiting state to the running state. This distribution format is called the initial distribution. The principle of the initial distribution is to evenly distribute the available real page numbers to the jobs chosen by the high level scheduling. If the requested virtual storage amount is less than the average, then only the virtual storage amount is given.

During the process of running a job, the page needed for real storage is dynamically varying. As long as the work batch of the job is ensured to stay in real storage, the efficiency of the system can be maintained without slipping. Therefore, one of the keys to the problem is to determine the size of the work batch of the running job at all times. We adopted an approximation method, which based on the page fault rate of the job predicts whether the work batch will expand or contract in order to determine whether to increase or decrease the number of real page dispatched for the job. The practical principle of page adjustment is that when the number of page fault exceeds $A \bullet \Delta t$ the number of real page distributed to the job is increased; where A is a constant which can also be adjusted, Δt is the difference of the present value of the job clock and value of the clock of the job when the previous page distribution took place. When the page fault number is zero and the distributed page number is greater than the actually occupied page number, the real page distribution number is reduced. When the page fault number of the job is between zero and $A \bullet \Delta t$ no adjustment on the real page number distribution is made.

When the main storage is overloaded and several running jobs all request increase of real page number dispatched, the system cannot satisfy the request of work group expansion for all the jobs. Then, the medium level schedule will order the job with the lowest priority number to release all the real storage occupied, to return that job to the waiting state from the running state. The use of this technique, in principle, can prevent the "bumpiness" phenomenon from occurring. When these jobs deprived of real storage once again are moved to the running state from the waiting state, the medium level scheduling does not process them according to the initial distribution. The distributed page number should be the deprived page number.

Whether it is an initial distribution or an adjustment of page number, the progress of the medium level scheduling issues a message to the corresponding job page management progress to notify the adjustments of real storage page. The page fault numbers, however, are provided by the page management progress.

The distribution of real storage by the medium level scheduling always begins with the running job with the highest priority number. The priority number of job is determined by the high level scheduling, based on the operating format of the job, the required virtual storage and the duration of running time. In addition, the job running priority number also increases with the time staying at the waiting or running state. This type of increase is made by the medium level scheduling. Furthermore, the operator can type in order from the control desk to appoint a certain off-line job as a priority job.

IV. THE MANAGEMENT OF PAGES

This system can provide to the users a maximum number of seven mutually "isolated" virtual spaces. In order to manage a virtual space, the system sets up a corresponding page management progress. Therefore, there are at most seven page management progresses whose major functions are as follows:

(1) Carrying out page fault treatment

The simplest request type of page transfer is adopted as the transfer principle of the virtual storage of the 013 machine, which means that only when the page visited by the program is not in the main storage (which also means that only when page fault occurs it is then switched to the operating system) that the page management progress is used to process the page fault. The treatment of a page fault can be simply described as the request of a real storage page, and to read the content of the page with the fault in auxiliary storage through the magnetic disc management progress, and followed by the correction of the page table and the corresponding term of the small table, and finally the liberation of the page fault progress.

(2) Scheduling the running of jobs with the medium level scheduling

The use of real storage by the page management progress is carried out under the control of the medium level scheduling. The medium level scheduling is responsible for the distribution and adjustment of the real storage for all the jobs. However, the page management progress provides such job running information as the real storage page number occupied by each job, the maximum occupied real storage page number and the page fault number during the two initiation periods of the medium level scheduling, etc., to the medium level scheduling, in order to facilitate the more reasonable scheduling of the use of real storage. In addition, it also provides some job running statistical data such as the total page number and the empty product at real storage.

(3) Processing the request for page table zone by high level scheduling

This request is unique to the system which is also considered as a weak spot. This is because in the design of the virtual storage hardware, for simplicity, it was decided that the page table zone of each job in the system is individually placed in a fixed location. These page table zones are not occupied by the system and are open to the users when used as the page table. Hence, when the high level scheduling selects a job from the back-up state to enter the waiting state and its corresponding page table zone is occupied by another job, it is necessary to reclaim the page table zone to establish the job. At this time,the high level scheduling progress issues a message to the corresponding page management progress to request the page management progress to forcefully take the page table zone back.

(4) <u>Taking back real storage when the job departs from the machine</u>

When a job departs normally, before termination and cancel of the page management progress, the job control progress in the same group issues a message to the page management progress to allow the page management progress to be responsible for the entire real storage page occupied by the job. When the job departs abnormally, this job is completed by the high level scheduling progress instead.

(5) <u>Explanation of the rule relevant to the real storage replacement</u>

The replacement rule of the real storage page is a more important scheduling factor. Its choice has a large effect on the efficiency of the entire system. A better replacement rule should be able to remove pages which are not going to be used in the near future from the main storage in time and at the right place. However, it should not remove the page which is going to be used in the near term. This means that a good replacement rule should be able to precisely reflect the dynamic expansion and contraction of the job work group. However, due to the numerous variations of the visited address space of the job on the machine, it is unpredictable. Therefore, the variation of different job work groups is not the same and the work group of the job also varies with the running of the job dynamically. Hence, it is very difficult to use a unified method to reflect the dynamic variation of the various job work group. For that, this system uses a simpler partial replacement rule. The measurement of work group variation is done by the medium level scheduling. The so-called partial replacement rule is to use the replacement counter (JS_{tt}) provided by the hardware and the usage of the main storage page to select the page which has not been used for the longest period of time (i.e., the maximum JS_{tt} value) in the real storage pages of the job to replace it.

V. ANALYSIS OF THE EFFICIENCY OF THE VIRTUAL STORAGE SYSTEM

The ratio of the efficiency η_v of a machine with a certain capacity of virtual storage and η_R which is the efficiency of a unit with identical capacity of real storage must be less than 1, i.e.,

$$\frac{\eta_v}{\eta_R} < 1$$

This result is apparent from the following two reasons:

1) the search in the associative storage may not be fruitful

2) when searching the page table, a page fault may occur which requires the changing of a disc.

There are multiple factors affecting the efficiency of the virtual storage. We conducted some experiments on the virtual storage efficiency problem. Here we performed some preliminary analysis on the following problems.

1. To properly select the line number of the fast page table storage in order to reach reasonable fast table hitting ratio.

The most intuitive method to increase the hitting ratio of the fast page table storage (fast table) is to increase the line number of the fast table. How it requires more equipment, how many lines of fast table must be chosen to obtain a reasonable hitting rate? If we use the probability estimation method, let us assume that the probability that the address of the current visiting storage is on the same page as that of the previous visit is 1/2 ; the probability of the current visiting address being on the same page as the one before the last one is 1/4; then the relation between the line number of the fast table n and the hitting ratio P is:

$$P = 1 - 2^{-n}$$

The corresponding relation between P and n is shown in Table 1.

TABLE 1. The relation between the hitting ratio of the fast table P and the line number of the fast table n

n	1	2	3	4	5	6	7	8
P	0.5	0.75	0.875	0.938	0.969	0.984	0.992	0.996

The 013 machine uses an eight line fast table. We used the monitor of the machine to carry out a statistical study on the hitting ratio of the running of various types of programs (compiled programs, BCY compiled program and the user objective program). The monitor gave the ratio of the actual table checking number k_1 and the hit number k_2:

$$P = k_2/k_1$$

The P values for various types of programs are shown in Table 2.

TABLE 2. The actual value of the fast table hitting ratio

type	compiled program	BCY compiled program	user program
P	∿98%	∿95%	∿90%

It should be pointed out that the various programs contained a certain amount of commands which do not require the transfer of address (e.g., change address operation, immediately accessible number operation, etc.). Through the monitor, it was found that this type of command was approximately 15-20%. Hence, the miss ratio of checking the fast table 1-P is very small.

2. To provide the necessary information to reduce the number of times in visiting the auxiliary storage.

In order to reduce the number in visiting the auxiliary storage, the address transfer mechanism of the 013 machine provides a correction indication digit. When it is operating in the data delivery mode for a certain virtual page, the hardware automatically changes the corresponding correction indication digit to 1 for the page table position. Hence, when all the real pages corresponding to the virtual pages with a correction indication digit 0 are replaced, it is not necessary to rewrite into auxiliary storage because there are copies of those pages in the auxiliary storage. The statistics for some problems indicated that because of the use of the correction indication digit it is possible to reduce the number of times in writing auxiliary storage by 30-40%.

3. Two important factors affecting the efficiency of the virtual storage.

The present representative equation to evaluate the efficiency of the virtual storage is:

$$\eta_V = \frac{\text{total processing time}}{\text{total processing time + page change waiting time}} = \frac{1}{1 + f(\omega)\,T_V/T_N}$$

where the total processing time is the amount of time required to complete a job if the virtual storage system is not used, T_V is the time required to visit the auxiliary storage once, T_N is the time required to visit the internal storage once. $f(\omega)$ represents the frequency of page transfer from auxiliary storage in the ω segment of the program:

$$f(\omega) = \frac{r}{t}$$

where r is the number of transfer from auxiliary storage during the page visit and t is the total number of page visit.

From the above equation, it can be found that $f(\omega)$ is related to the program characteristics and the ratio of virtual page number and real page number V/R. When $V/R \leq 1$, then $f(\omega) = 0$, $\eta_V = 1$; when $V/R > 1$, then the value of $f(\omega)$ increases, η_V value decreases. The larger the value of V/R then the smaller η_V becomes. Hence, the virtual vs. real ratio V/R is an important factor affecting the efficiency of the 3 virtual storage. In addition, it can also be found from the formula that T_V/T_N is another important factor affecting the efficiency of the virtual storage system. The smaller the value of T_V/T_N, the higher the value of η_V becomes; the higher the value of T_V/T_N is, the lower the value of η_V becomes. Now let us first discuss the effect of T_V/T_N on η_V.

The parameter T_v/T_N is determined by the hardware. In order to reduce T_v/T_N, in some computers a multi-level storage system is used. Between the slower auxiliary storage and the faster main storage some middle transition levels are added so that the efficiency of information transfer between the internal and external storage systems can be improved. The technical standard of the preseng magnetic disc in the 013 machine is low, $T_v/T_N \approx 10^6$. Under this condition, the efficiency of the virtual storage system established is indeed low. This point has already been apparent from the preliminary operation of the machine. Just because the value of T_v/T_N has such a large effect on η_v, people have the misunderstanding that when the T_v/T_N value is high a virtual storage system cannot be established. We believe that such a conclusion is biased to some extent. This is because the η_v described above is obtained through a comparison between time spent to process a job using the virtual storage system and that to process the same job using a real storage system of the same size. This comparison was carried out using the machines with different hardware equipment. We believe that in designing a machine the visiting cycle of the auxiliary storage and the main storage should be a given condition. If this condition does not change and if the virtual storage system is not used, then when the occupied space of a job is greater than the capacity of the real storage,the user must have a more complicated coverage processing technique to manually materialize the information exchange between the main storage and the auxiliary storage. However, this exchange is similarly affected by T_v/T_N. Therefore, when discussing the effect of T_v/T_N on the efficiency of the virtual storage system, it has to be stressed that the comparison should be carried out under identical conditions; i.e., a comparison must be made with regard to the times spent on the same job using manual scheduling and automatic scheduling (virtual storage) under the conditions of identical main storage capacity and visiting cycle as well as identical auxiliary storage capacity and visiting cycle.

We conducted such an experiment using both the manual scheduling and the automatic scheduling methods to compute the same problem in order to compare the time to process the same job based on two different methods to evaluate the efficiency of the virtual storage system.

The computational process of the problem is to first form a 500,000 element matrix and then to carry out computation for the above matrix using an element elimination method. In addition, the lateral cumulative sum of the matrix is obtained. After the element elimination calculation, the cumulative source of the matrix in the longitudinal direction is obtained and compared with the transverse cumulative sum to indicate whether the results of the computation is correct.

During manual scheduling, due to the limitation of the main storage, the matrix was divided into several submatrices and stored in the disc. They were transferred into internal storage for processing, and then re-connected together. The program used the characteristic that the magnetic disc transfer and computation can be carried out simultaneously.

During automatic scheduling, because the virtual space is one million characters, the user flatly expanded the original data. After expansion, the element elimination started from the beginning and the lateral sum was added. This computational method required the repeated transfer of data from beginning to end. The localization of the program is poor. Hence, the condition is stringent with respect to the evaluation of the efficiency of the virtual storage system.

The experimental results showed that to complete this job using manual scheduling took eight months while it took 17.5 minutes to do it with automatic scheduling. From these experiments, it was found that under the conditions of identical hardware, even though the technical standard of the magnetic disc is poor, the use of virtual storage will require longer job running time than manual scheduling. However, it is too long for the users to bear.

Now let us discuss the relation between the virtual vs. real ratio and the efficiency of the virtual storage system. The V/R value 35
gives the ratio of the virtual storage capacity vs. the real storage capacity of a job. It is also called the virtual/real ratio. If $V/R < 1$, it represents that the main storage is larger than the needed virtual storage and it is not necessary to use the information in the

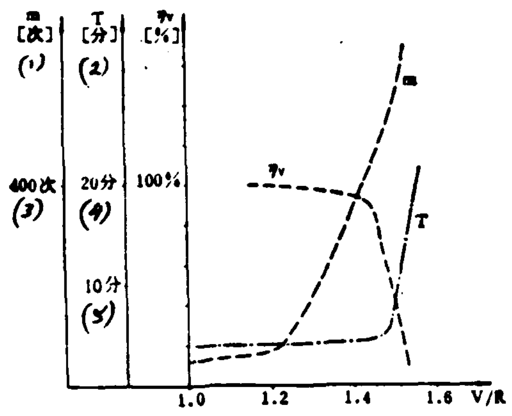


Figure 2. The relation between V/R and m, T, η_V
1--number; 2--minute; 3--400 times; 4--20 minutes; 5--5 minutes

auxiliary storage. Therefore, the $\eta_V = 1$ at this time. If $V/R > 1$, it represents that some of the pages of the job are not in the main storage which must be transferred into the system from auxiliary storage. At this time, $\eta_V < 1$. The higher the value V/R becomes, the more frequent the number of auxiliary storage visits becomes and the smaller η_V is. When $V/R >> 1$, due to the excessive number of disc changes, the efficiency of the virtual storage system is drastically reduced. It must be pointed out that the disc change number and the V/R ratio are far from direct proportionality; it is related to the running characteristics of the problem. The same virtual vs. real ratio for various programs may cause a huge difference in the disc changing times.

Specifically with respect to the relationship between the V/R value and the efficiency of the virtual storage, we performed some experiments. The relationships of the V/R value of a problem running on the 013 machine with the number of auxiliary storage visits m, the time to solve the problem T and the efficiency with virtual stage η_V (assuming when $V/R = 1$, $\eta_V = 1$) are shown in Figure 2.

From Figure 2, it can be found that there was a "turning point" in the corresponding relation between V/R and η_V. The value of the turning point is greater than 1.

1) When the V/R value is smaller than the turning point, η_V approaches 1. This shows that the virtual storage system can indeed provide the users a storage space larger than the real storage space under the assurance of the efficiency of the processor.

When the V/R value is greater than the turning point, the [illegible] have significant pages not in the main storage. Therefore, the number of auxiliary storage visits increases and η_V drastically decreases. This is another important characteristic of the virtual storage system. It should be pointed out that the position of the turning point is different for different problems. The position depends on the size of dynamic variation of the work group of the problem. Therefore, it is necessary to ask the users to have a correct understanding toward the virtual storage space. If a job has a larger virtual storage space but the localization of the program is better, then it is still possible to run efficiently. On the contrary, if the users assume that the relation between time and space remains the same as using the main storage in the past after using the virtual storage system, then it is highly probable that the time-space relationship of the virtual storage system may be destroyed to cause an apparent decrease in the efficiency of the virtual storage system.

VI. CONCLUSIONS

The virtual storage system we materialized in our institute is the first one. It is also one of the earlier ones in the country. Due to lack of experience, an attempt was made to reduce the equipment to the extent possible to realize the virtual storage system from the hardware point of view. Therefore, its function is not perfect. There remain inconveniences in the software conditions. From the point of view of the software, due to the limitation of the hardware, the most obvious problem is the supply and demand mismatch problem of the page space. The virtual storage system established on this basis has low efficiency. The experiments conducted with respect to the analysis of the efficiency of the virtual storage system were still coarse and need further refinement.

It should be pointed out that Comrade Li Shuyei has done a great amount of work on the analysis of the efficiency of the 013 machine during and after the process of realizing the virtual storage.

REFERENCES

[1] Lin Kuoyen, Li Shuyei, Fan, Ben-kui, "With regard to the virtual storage system of a machine", Status of Electronic Computers, 1978, 12.

INPUT/OUTPUT MANAGEMENT OF CZXT-013

361

Li Zhi-cneu, Gu Yu-Qing, Zheng Li-jun and Zhao Wen-qin
(Institute of Computer Technology, Academia Sinica)

ABSTRACT

The input/output management of the CZXT-013 machine is divided into two parts, i.e., the logical device management and the physical device management. This article emphasizes the introduction of the logical device, format processing and device management and their corresponding characteristics.

I. INTRODUCTION

The input/output devices of the 013 machine consist of four magnetic discs, eight magnetic tape drives, four photoelectric paper tape input devices, four wide width printers, two paper tape punches, two keyboard symbol displays (user display, monitor display), one graphic display and one electrostatic printer. With the exception that the magnetic discs are working under the control of the magnetic disc channels, the others work under the control of the exchanger channels.

The major tasks of the input/output management are:

(1) to provide the users with a simple,clean,good environment convenient in use, simple and flexible in operation and safe and reliable. This system realized these requirements through the concept of leading logical devices;

(2) to carry out format processing according to the user job and the internal input/output requirements of the system,to complete the information exchange inside and outside the machine,to facilitate communication between man and machine.

(3) to actually manage the physical devices in order to effectively use the hardware resources.

The following is an introduction to some of the considerations and corresponding characteristics of the three areas.

II. THE USE OF LOGICAL DEVICES

In order to make the user program relatively independent of the actual physical devices, in order not to cause a great difference in use due to the different characteristics of the actual external devices, and also in order not to change the user program because of changes in the external equipment, we brought in the logical device concept as the virtual expansion and reform of the actual physical devices. The logical devices we introduced have the following characteristics:

(1) Each logical device is expressed by a device number. Each user job can use 32 device symbols, the numbers are from 0-31. Thus, the generalized input/output commands or the keyboard instructions in the user program have a unified format so that it is easy to memorize and easy to use.

(2) The use of the logical device is similar to the use of document which means before the use (read, write, transfer, etc.), it is necessary to carry out a definition step through the open command in order to determine the device type. After use, it is released through the close command. Obviously, the closed device symbol can be re-

This paper was received on August 5, 1980 362

opened if necessary (notice that any type of device can be defined at this time).

(3) The definition and release of the logical device and the reading and writing of the logical device are carried out independently. In general, the former is carried out through keyboard commands and the operating manual commands, while the latter is directly arranged in the user program according to the requirements. Thus, before use, it

is possible to change the device type at any time through the keyboard command or by modifying the operating manual. It is not necessary to change the user program itself. Therefore, a certain extent of flexibility and reliability can be assured.

(4) Due to the fact that it is expressed by four digits, the type of the logical device can include 16 different types. This facilitates the expansion of new types of external devices. Some of the main device types are:

(1) A certain physical external device (e.g., on-line printing and on-line hole punching, etc.);

(2) a combination of several devices (e.g., several paper tapes to form a paper tape document);

(3) a group of information on the magnetic disc or magnetic tape (e.g., input document, output document, disc document, magnetic tape document, etc.);

(4) the transfer between devices (transfer between the discs and tape devices or transfer between the disc/tape device with the printer and hole puncher);

(5) expansion type device: the system, in order to facilitate the expansion of new types of external devices, designs this type. However, the users are still provided with the pressurized code symbol of the corresponding device and the system centrally converts into the expansion type device. The expression type device is expressed by the progress number of the device management progress. The keyboard symbol display and the optical pen graphic display of the system both belong to the expansion type devices. The expansion type can correspond to different types of output devices simultaneously (for example, the printer and the display can be simultaneously used as the output).

(5) the management progress relevant to the logical device belongs to the job family where the index progress is responsible for the definition and release of the logical device. The read progress and the write progress are responsible for the format processing of the reading (input) and writing (output) of the logical device, respectively. The basic document progress is responsible for the checking of the type of the logical device, the establishing of relationship between the logical device and the actual device management progress (according to the type) and the transfer control of the read/write printer management and the device. It is the actual connection between the job and the system;

(6) the description of the logical device is realized through two tables. One is the user file display (UFD) and the other one is the partial user file display (PUFD). The length of the UFD table is 32 units. Each unit corresponds to a device number which is used to record the type of the logical device, the relevant read/write characteristics and the pointer of the PUFD table, etc. The length and the structure of the PUFD vary with different devices. Some devices (such as the on-line printer, on-line hole punch and transfer and expansion types, etc.) do not have a PUFD which primarily record the read/write pointer of the actual device and its manual;

(7) the errors which occur when using the logical device are reported to the corresponding job management progress which then requests the writing document monitor progress of the corresponding job to enter the erroneous message into the monitor document. When leaving the machine, it is printed on the printer. For on-line jobs, it can be sent out of the machine at any time.

III. PROCESSINGS OF FORMAT

During the input/output process, if the information remains unchanged, it is called a formatless transfer. On the contrary, it is called format processing reading or writing. The considerations regarding format processing in this system are:

(1) the paper tape is the main method for the input of the original data and programs from the user. In addition to the processing of information according to the format table to enter the input table, for the convenience of the users a simple and intuitive set of paper tape input format language has been specially designed to allow the users to write the paper tape message in their customary way.

The paper tape input format language is divided into information format and control format. Before each sentence there is a corresponding <format symbol>. For the information format, the format symbol 36 indicates the type of the information (such as integer number, floating number, octal number, symbol series and command, etc.). The series of information which follows is basically expressed according to the customary way. For example, with the exception of a symbol series, the information is separated by the sign ; . For the control format, the format symbol represents the control type which is followed by the corresponding parameters (such as the address of processed information, repetition number of the information, the automatic starting address after the paper tape output, etc.);

(2) a basic format character system has been designed primarily to compile printing or display the output information,but it is also applicable to other output devices and even possible to store back to the internal storage the output information after input processing using the same format table (except for printing and display output). The closed loop characteristic of input/output has been realized.

The user program can compile the needed corresponding format chain during output or directly use the regular format code (the pre-compiled format chain in the system) provided by the system. The address of the output information is usually given by the address chain. Each address character in the address chain points out the internal memory address of the output information and its length. The address of the output information can also be given by the basic format character (instant address or instant symbol).

The basic format character occupies an internal storage element which is essentially divided into information format and control format,each with 32 types of formats. The information format indicates the type, width, significant figure and information source of the output information. The control format indicates the compilation form (blank row, blank space, position fixing, rotary printing, column printing, etc.) of the output information, the motion control of the format chain (such as format transformation) format repetition, format rotation, format return, etc.), and other format explanations.

This set of basic format system has the following characteristics:

(1) it allows the insertion of a segment of programs in the format chain or address chain to change the direction of motion of the format or the address chain by the result of the running of the program which provides some flexible method for editing relying on the input/output character chain;

(2) the output format processing is carried out in two steps. The first step uses a unified processing procedure to carry out the processing (semi-processing) to produce the intermediate result. The characteristic of this processing step is that it does not depend on the device type. The intermediate result of the semi-processing step has a standard model which is suited for any device management progress. Most of the format processing is accomplished in this step in which the information format transforms the internal storage information into output symbols and some control formats are transformed into the corresponding format explanation sentences. The second step of processing is carried out independently by the device management progress which transforms the standard intermediate results into the output results of the various corresponding devices. The step of processing usually has different interpretation with various devices. Some devices (such as magnetic discs, magnetic tapes, etc.) directly use the intermediate results as the output results (discs and tapes). Some devices (such as printer, display, and electrostatic printer) must transform the format explanation sentences into output symbols

to complete the processing job. The paper tape hole puncher only performs some analysis on the intermediate result and punches holes in the paper tape as the output;

(3) the semi-processed intermediate results are described by the semi-processed format language whose form is as follows:

YC <alphabet> <explanation> YR

when YC is the exit symbol whose code is (8) 17; YR is the enter symbol whose code is (8) 16; <alphabet> is a format explanation type; <explanation> is a series of characters expressing the content of the explanation and parameters.

In this set of semi-processed format language, with the exception of the symbols YC and YR, each character symbol belongs to the region of symbols which are suitable for output in order to facilitate the printing and hole punching for the convenience of independent testing;

(4) the intermediate results directly recorded on the disc, tape (magnetic) and hole punched paper tape can be processed on the reverse direction by reading to return the semi-processed results to the original internal storage information to realize the close loop characteristics of read/write;

(5) due to the lack of experience, there are still many shortcomings in the semi-processed intermediate language such as the produced intermediate results are still very long. However, the separation of format processing still has its advantages especially when the device itself has some processing capability. At this time, the second step of processing can be accomplished entirely by the device;

364

(6) the communication information with regard to the job group in the system is processed by the writing monitor document progress. The provided format system is the basic format character system whose processing type is similar to the one in the writing progress. The communication information relevant to the system group in the program of the system, especially the breakdown notification information of the device management progress whose position is placed on top of the

level of the writing monitor document progress, can no longer be processed by the writing monitor document progress. It can only directly notify the monitor display output progress and breakdown printing progress to request for an output notification. At this time, a predetermined format is used to reflect all the information and its corresponding format in the message.

VI DEVICE MANAGEMENT

Basically, this system has three types of devices: virtual device (input/output well), magnetic disc device and exchanger devices (such as magnetic tape, paper tape input, printer, hole puncher and display, etc.).

1. The management of a virtual device

In order to raise the efficiency of the system and to reduce the cycles in problem solving, the system provides an off-line input/output form .pseudo off-line input/output form) in addition to the on-line input/output form.

The so-called pseudo off-line input/output form is to leave a certain amount of space (approximately the capacity of one disc) on the magnetic discs as the input/output well. The user can input paper tape information through the escape entering command into the well to form a well-entered document. When the job is executing a sentence, it can be directly read from the well. Similarly, the output information can also be recorded in the well. When necessary, by going through certain commands or when departing from the machine, the well output document can be unloaded to an output device (printer, hole puncher, electrostatic printer, etc.). The advantage of the pseudo off-line input/output form is that it smoothes the input/output flow to replace the slow speed special equipment by the fast speed shared device for convenience and high utilization rate of the devices.

The management of virtual device is mainly the distribution and management of the well zones. The system has a well zone distribution and exchange progress and a well zone inquiry and retrieval progress to be responsible for the distribution, exchange, inquiry and retrieval of the well zone. The connection in the well zones is the implementation of the well address chain. The lead of the well address chain establish some relation with the corresponding device symbol.

2. The mangement of the magnetic disc device

This system is equipped with four magnetic discs; correspondingly there are four disc management progress to be responsible for the management. Because the magnetic discs are used not only as the auxiliary storage of the virtual storage and the input/output well of the system, but also it opens a certain amount of space directly to the user programs as external storage (including archive documents); therefore, the magnetic discs have a unique address in the system. But because the moving speed of the magnetic head arm of the magnetic disc is slow, in order to reduce the number of times of arm lifting and the movement distance to the extent possible, the system especially sets up a disc read/write control progress to be responsible for the optimization of arm lifting. The optimization method used now is the scanning optimization method,which lines up all the read/write requests according to the seouence,and the rule to respond to a read/write request is that the closed zone to the forward direction of the motion of the magnetic head arm gets processed first. Thus, the magnetic head arms sequentially process the read/write requests until there is no more request in the forward direction, then it changes direction and begins to process the read/write requests similarly. It scans back and forth until all the requests in line are processed. This optimization process is simple and highly efficient. Each read/write request can obtains a response in time.

3. The management of the exchanger device

This type of device is the main input/output device of the 013 machine. Its obvious characteristic is that it is directly related to the operators or users. The management progress related to this type of device is divided by its usage. The same device can be shared by many progresses. As a few examples, the paper tape input device is divided into the paper tape lead progress, the paper tape linkage progress and the paper tape escape entering progress; the printing output device is divided into the on-line printing progress, the off-line printing progress and the breakdown printing progress; the display device is divided into the display input progress and display output progress, etc. In order to handle the competition for the external device resources by several progresses, the system gives each subchannel and each device with a corresponding signal quantity (its 365 initial value is 1). When a certain progress needs a certain subchannel of a device, it is ruled that the u.age right to the device is determined first and the right to use the subchannel is decided later. Only when permission is given, then it is allowed to be used. Once the permission is granted, it practically is exclusive to the progress before the release. When the usage reaches a stage (such as printing one line, reading the paper tape into a buffer zone) or the usage is complete, the right to the subchannel is released first in order to free the subchannel and other device channels and finally the right to the device is released. This type of competition and release are realized through the operation of the corresponding signal quantity P or V.

The operation of this device often requires people to accomplish (such as the loading and unloading of the magnetic tape and paper tape, the changing of printing paper, electrostatic paper and the paper tape and the transmission of the display keyboard command). Therefore, when designing the function in use, it is not only necessary to consider the highly efficient usage of the devices, but also the convenience of operation by people and ease of identification.

The basic function of this device management progress is to be responsible for the distribution of the actual equipment, to formulate the channel control character chain, to start or to stop the device and to process the interruption of the corresponding subchannel and to interfere with the interruption. When the device breaks down, it carries out breakdown notification and processes the replacement of the broken down device.

The distribution of this type of device varies with the device. The selection of the magnetic tape drive and the paper tape machine is resolved by the coordination of the users. The users choose the usable equipment directly and then reaction is established between the magnetic tape number or the paper tape name and the corresponding unit number. Some devices (e.g., user display, optical pen graphic display, etc.) are fixed for on-line job use. Some of the devices are fixed for off-line output use (e.g., the electrostatic printer, etc.). As for the other devices, when the number of the usable devices are relatively adequate,the static distribution is used to the extent possible to allow the on-line job to enjoy some of the devices (such as the on-line printer). Only when the usable devices are not sufficient is the dynamic distribution used. At one moment, the on-line form is used and in another moment, the off-line form is used. In this case, we must consider the completeness and identification of the output information.

Due to the special characteristics of the devices, the management progress of each device has its own special considerations. For example, in the case of paper tape input, we must consider the selection of the information segment name and the processing of changing disc across a drive unit. For display output, we must consider the division of the display screen. For the magnetic tape, we have to consider the tracking and checking of the position of the magnetic head. For printer and display output, we have to consider the processing of the pre-determined format.

V. CONCLUSIONS

The system has taken into consideration the irrelevance of the device; therefore, the structural layers are more clear, its reliability is high and its adaptability is strong. However, because the completion of one read or write request must require the transfer of relevant read or write messages between several progresses (single initial read or write message, multiple continuing read or write message, single terminal read or write message), the message must be processed twice. The adopted method is the single buffer technique (512 elements); therefore, the transmission efficiency is low and the processing time is long. In addition, these messages are transferred between the jobs and the system. If some jobs require to carry out real time interference (such as the restarting of the job, the temporary halting of the input/output operation, etc.) or require the waiting of the return message between different groups, the response is slow and the realization is difficult.

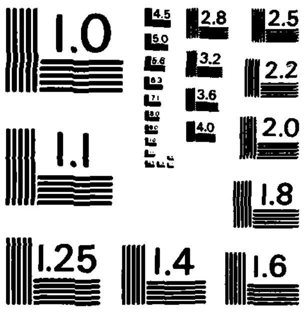
In summary, the system needs to be under operation for a period of time to discover problems in order to further perfect and improve the system.

AD-A122 743 JOURNAL OF COMPUTERS (SELECTED ARTICLES)(U) FOREIGN TECHNOLOGY DIV WRIGHT-PATTERSON AFB OH G LIU ET AL. 14 OCT 82 FTD-ID(RS)T-0369-82

2/2

UNCLASSIFIED F/G 9/2 NL

END
DATE
FILMED
2 83
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

CZXT-013 TESTING AND DEBUGGING

Liu Guo-hen and Cai Chun-lei
(Institute of Computer Technology, Academia Sinica)

ABSTRACT

In this paper the valuation between the reliability and testing of the operating system is first discussed. Then, the emphasis is placed on the introduction of the testing method of the operating system of the 013 machine and the two testing tools, the process tester and the system monitor, which are used in the unit testing phase and the integration testing phase, respectively. The process tester is a common testing program for the testing of the process module block. The system monitor is an information collection and output program added on to the system control center.

The operating system is large scale system software; its accuracy and other software is affected by factors such as its scale, complexity and the standard and experience of the developing personnel. Most probably it cannot be assured to be error free. In addition, the characteristics of the operating system itself also have some negative effects on its accuracy. One reason is that the system is concurrent, which means that the system has the characteristic of handling many things simultaneously. Another reason is that the system cannot be described in any form,which often causes the description of a problem to be vague and unprecise. These factors all have a negative effect on the accuracy of the system in which the concurrency of the system has the most influence on the accuracy of the system. It, on one hand, makes the system use a working format which is not customary to people--serially and sequentially considering and processing the problems. This greatly increases the complexity of the program.

On the other hand, due to parallel working, the processing of various jobs is a synchronous which may cause errors related to time. The frequency of this type of error is low and the conditions to reappear is very difficult to re-establish. Therefore, it brings great difficulties to the tesing and correction of the program.

At the present moment, the accuracy of the program of the entire operating system cannot be formally or informally verified. It is also not possible to check out by an automatic technique. Under these conditions, as a method to discover and correct program errors, testing is the basic important technique to assure the reliability of the system. Hence, the selection of the testing method, testing procedure and testing tool determine the assurance of the reliability of the system. If the proper selection is made, then testing is fast and thorough which saves manpower and materials. Most of the errors are eliminated in the testing stage. No more or very few defects remain in the system. If,on the contrary,it not only wastes manpower and materials, but also the reliability of the system is hard to assure.

I. TESTING METHOD

When selecting the testing method for the operating system of the 013 machine, we considered the structural characteristics of the system and chose a specific testing method and determined the testing procedure and equipped with the testing tool. The CZXT-013 is a layered system using the process module block as the major part. The system has eight layers including the nucleus. Each layer is further divided into several levels. The nucleus layer provides the process running environment. Program modules are used for running. The other layers have their program modules in the process module blocks of the process running form. The entire system has approximately 30 process

this paper was received on August 5, 1980

module blocks and they occupy about 90% of the program. Therefore, 36
the testing of the process module block is the major task of system testing.

With the exception of the interruption processing program in the nucleus layer of the system, the programs are all original language processing program module blocks. The functions of these module blocks are independent, the connections are standard and the module blocks are related according to the layer relation. The amount of programming in each module block is small. On the average there are over 60 instructions. Therefore, the testing of the nucleus layer program uses a testing technique which is compatible with the structure by layer and block. There is no special testing tool needed.

The large amount of process module blocks in the main body of the system has the following characteristics. The first is that the functions of these blocks are independent and complete with a certain degree of independent running capability. Hence, if the proper initial environment of the process operation is created, they can be tested individually. Secondly, these processes collaborate with one another and among them there is a certain constraint relation which is realized through testing with various original languages (such as control language, resource language, communication language,,etc.). Thus, in order to carry out individual independent testing with each process module block, in addition to the need for an initial environment, the proper external conditions are also required. These external conditions are concentrated in the original language of the testing results.

With respect to these characteristics, we used different testing methods and testing tools in the unit testing phase and the integration testing phase. In the unit testing phase, we designed a common testing program--the process testor to carry out testing with each process module block separately. Because each module block was compiled differently with various progress, there is a difference in the expected time. Therefore, the unit testing sequence does not follow the layer sequence. Once a block is prepared, it is tested. It is also possible to test a number of blocks alternately. In the integration testing phase, we designed a system monitor to be attached to the nucleus layer program to monitor the integration testing process. The process module blocks which have been unit tested are sequentially brought into the system,using the nucleus layer program as the center. When a process module block enters the system to be integration tested, the collaborating process module blocks, especially the inner layer process module blocks which provide service functions to it, have entered the system and performed integration testing. Thus, the testing personnel can concentrate effort on the tested module block.

Basically, no effort will be spent on its operating environment.

II. UNIT TESTING TOOL--THE PROCESS TESTOR

The process testor is a testing tool used to carry out unit testing in the CZXT-013 machine,which is a common testing program. The purpose of its establishment is to reduce the number of testing programs to the extent possible under the assurance of testing quality to facilitate the testing of every process module block. The process testor has the following functions:

1. It simulates all the language capabilities of nucleus layer. The real capability of the original language in the nucleus layer is not suitable for unit testing use, therefore, it needs qualification and simulation such as the process control language, etc.

2. It provides initial environment to the operation of the processes. If necessary, it is possible to provide the unique initial environment for the operation of the process according to the initial reserve program given by the testing personnel.

3. It uses the testing data provided by the testing personnel to simulate all the activities of the collaborative processes in order to establish the dynamic working environment for the tested process, such as the simulation of information exchange between processes, the values of the commonly shared data zone or buffer zone and the simulation of various functions of the external devices.

4. In addition to the simulation of the communication language, it is possible to compare and verify the information originated in the process with the expected one.

5. It has an output function. It can print out the testing language, process message, process and work space contents of the tested process.

6. It can process the attached "observation" language attached to the process testor. The user can place the "observation" language in the areas of interest in the testing program. When the program is executed to the observation point, the testor can output the data of interest.

The working principle of the process testor is briefly described: The testor carries out the usual preparation work for the tested process operation, such as clearing the storage and preparing for the input of language simulation program, etc. Then the initial preparation program (this is the only program to be compiled) prepared by the testing personnel provides the working parameters such as measured data position, process working area position and so on to the tester. If there are unique requirements to the initial environment of the process, they are also prepared by this program. Afterwards, it is shifted to the tested process. This means the beginning of testing for the establishment and starting of the tested process. In the execution of the process under testing, whenever an original language was executed, the control would switch to the corresponding simulation program of the tester. At this time, output of the original language name, position and parameters can be obtained. It also compares the parameters of the language with the testing data provided by the testing personnel and the results are also available from the output. After the output, it begins the simulation of original language function. If the original language has a return message, then the provided testing parameters are returned to the tested process. At this point, an original language simulation is complete. The control returns to the tested process to continue the execution. It is repeated until all the testing data provided by the testing personnel are completely used.

The use of the process testor to carry out the process module blocks can minimize the work in the compilation of the testing program to the extent possible and the effort of the testing personnel can be placed in the selection of testing data so that the program testing is more thorough. The tester can automatically and continuously change the testing data and verify with the testing results to further automate the testing process. Thus, it not only does not require the testing personnel to compile the testing program, but also avoids the manual operation of the testing process. The tester collects, arranges and outputs the input testing data and the testing results produced during the testing process according to the requirements of the testing personnel,automatically thus avoiding the omission of any relevant information due to lack of consideration and operational error in the manual testing,which is also beneficial to the analysis of the testing results afterwards. The process module blocks do not have to be altered during testing by the tester. The qualified module blocks after unit testing can enter the system to carry out integration,which makes it possible to avoid the complicated procedures and possible error by the testing of a program from a running state to a testing state,and then back from the testing state to the running state.

Test results showed that the application of the process tester has reached the projected objective. It basically avoided the compilation of testing program under the provision of assurance of the testing quality to save the testing time and to speed up the testing progress. For those module blocks with complicated relations with the external and a lot of programming, the results were more obvious.

III. INTEGRATION TESTING TOOL--SYSTEM MONITOR

The purpose of carrying out an integration testing of the system is to check and verify the dynamic constraints between the processes and the accuracy of the information exchange. In such a highly developed system, it is very difficult to determine the breakdown situation and judge the source of error without an effective integration testing tool. The structure of the system adopted some standardizing procedures with respect to the module blocks. The interaction between an independent component process and the environment (the system and its components) of the system is realized by the original language. The use of original language must go through the two connecting programs--the inlet module block and the exit module block. In order to facilitate system integration testing, we specifically installed a system monitor to effectively monitor the running of the system and to reflect the collaborative processes and the information exchange situation.

The system monitor is a set of information collection and output programs attached to the system control center which is the process scheduling module block and the inlet/outlet module block. It can collect and output the following information:

1. The creation, extinction and most of the state transition of the processes

2. the access to the central processor by various processes

3. the communication and relevant information between processes

4. the access to storage resources by the various processes

5. the use of the system by the user programs

6. the use of external devices of the processes and their corresponding channel programs

7. if necessary, the monitoring of the alternate use of the module blocks in the nucleus layer.

Due to the fact that the monitor is attached to the central control of the system, as long as the system is running the monitor is effective. In order to facilitate the operation, the start and stop of the monitor and the monitoring of specific process or all the processes can be controlled through the monitoring desk. The system monitor does not affect the normal operation of the system except for the "clock" of the system. It does not influence the interaction between the processes. The monitor itself is a close looped system. It has its own independent function which does not rely on the system being tested.

In the integration testing period, in addition to the use of the system monitor, some simple and routine testing tools, such as the random printing of the process scene program, the sampling of the internal storage element and the random printing of the program, etc., are used as auxiliary methods. The random printing of the process scene program can be carried out based on the program or at the request of the control desk under the condition that it not affect the work of the system. The storage content, work space content especially used by the process, the buffer zone content and the information related to control in the process control block of the specified process are printed.

The use of the system monitor brought a great deal of convenience to integration testing. The usual connecting problems between process module blocks can be identified or show obvious clues through the use of the system monitor. Because the most important and commonly used communicative form is the information buffered communication , the system monitor has a function to monitor the communications. The monitoring of messages can more objectively reflect the sender, the sender position, the receiver and the content of the information. This is very beneficial to the judgment of the nature of the error, erroneous process and erroneous position.

As a testing tool, it is required to be simple by itself to reduce change to the system to the extent possible. Therefore, the monitor cannot have the function to go backward. It simultaneously releases the collection and processes the output. Therefore, it has the shortcoming of large output. For errors occurring after the system has been running for a while, the address coincidence stop method is used to get near the erroneous region. Then the monitor is started to reduce the output information. To give the monitor the added time reversal function will be beneficial to the discovery of time-related errors, to reduce interference to the "clock", to compress the output information and to further improve the effect of the monitor.

The process testor and the system monitor are very effective in the testing of the system. They are very satisfactory testing tools. After the unit testing, the process testor basically has accomplished its historical mission and is no longer used. The system monitor and other testing tools still remain in the system in the trial computation and maintenance process so that causes of problems can be found immediately to rapidly eliminate breakdowns.

E

ED

83